SEGMENTATION OF IMAGES WITH SEPARATING LAYERS BY FUZZY C-MEANS AND CONVEX OPTIMIZATION

B. SHAFEI AND G. STEIDL

Abstract. This paper is concerned with the segmentation of two- and three-dimensional images containing separated layers. We tackle this problem by combining the fuzzy c-means algorithm with recently developed convex multi-class segmentation algorithms, where we modify the data term of the corresponding functional to involve the information of the layer structure. We solve the optimization problem numerically by applying an alternating direction method of multipliers in conjunction with the fast discrete cosine transform to solve the involved linear system of equations. We demonstrate the performance of our method on synthetic and real-world images. In particular we deal with the segmentation of three-dimensional images arising from micro-computed tomography of C/SiC-ceramics by synchrotron radiation.

1. Introduction

Segmentation is a fundamental task in image processing which plays in particular a role in many preprocessing stages. In this paper, we are concerned with the segmentation of special images containing separating layers. Our aim is to incorporate the knowledge about the layer structure into the mathematical segmentation model. Our original motivation to consider such problems comes from the segmentation of three-dimensional image data arising from the micro-computed tomography of C/SiC-ceramics by synchrotron radiation. Since carbon fiber reinforced silicon carbide (C/SiC) ceramics can withstand extremely high temperatures and is tough with respect to fractures, these lightweight long lasting materials are used in jet engines, termal protection systems of space-crafts, brakes in passenger cars, friction bearings and lightweight gears. The C/SiC-ceramic data examined in this paper was produced by the liquid silicon infiltration process. This process starts with a C/C preform, where carbon fibers are bounded by low-density carbon and subsequently infiltrates the preforms by liquid silicon which reacts with the low-density carbon forming a layer of SiC. The development of efficient production methods requires tools to monitor the quality of this siliconization process. The segmentation of the microstructural C/SiC-ceramics data obtained by micro-computed tomography can serve as preprocessing step here. Fig. 1 shows a part of one slice of the micro-computed tomography data of C/SiC-ceramics. Due to the imaging process it is difficult to see even for the human eye that the SiC-layer separates carbon from silicon everywhere. Therefore the segmentation of the C/SiC-ceramics data is a challenging task. Simple thresholding techniques fail due to the facts that different segments can still contain the same gray values and that the separation by the SiC-layer is not clearly visible everywhere.

In this paper we propose to segment images with separating layers by modifying recently proposed convex relaxation methods for image multi-labeling [1, 17, 18, 23, 29] with respect to the layer structure. More precisely, our aim is twofold:

Date: June 19, 2011.

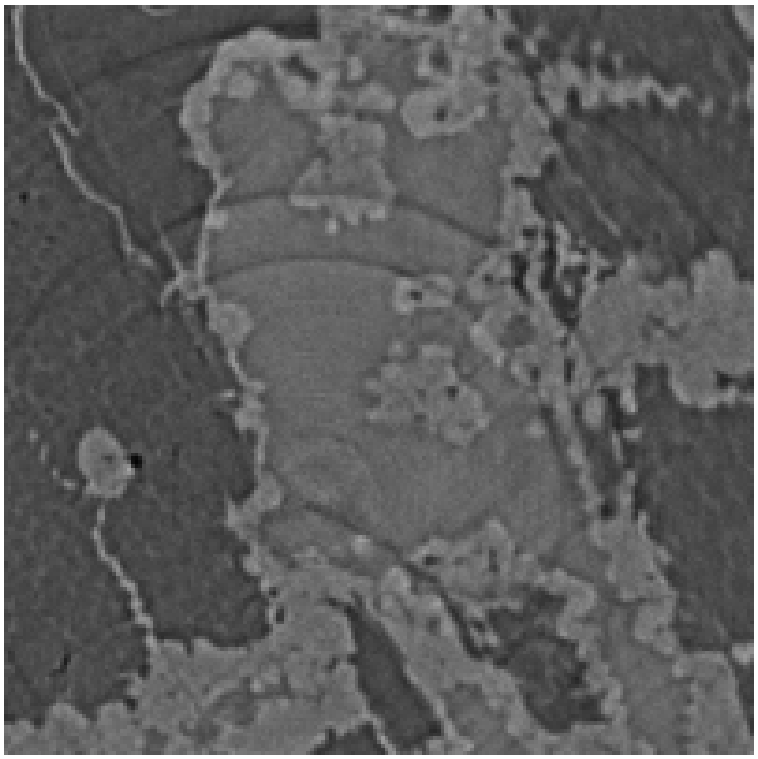


FIGURE 1. Zoom into one slice of 3d micro-computed tomography data of C/SiC-ceramics. Due to the imaging process there appear artefacts, e.g., rings, and similar gray values make it very difficult to distinguish between the different layers.

- Combination of the fuzzy c-means algorithm with a TV-regularized convex optimization model. Both algorithms intend to find instead of a labeling function a label defining matrix as minimizer of a certain functional.
- Incorporation of the layer information into the data term of the convex model.

We compute the segment prototypes by the fuzzy c-means algorithm (FCM). Then we use these prototypes together with the separating layer information in the data term of a convex optimization model which regularization term is the TV-functional. For solving the resulting convex optimization problem we propose an alternating direction method of multipliers (ADMM) which is for this problem identical with the alternating split Bregman algorithm [13] and the Douglas-Rachford-Splitting algorithm [8, 10, 11, 27]. We focus on a discrete model which uses matrix-vector notation based on tensor products of matrices. In the numerical part we demonstrate the good performance of our algorithm for 2d and 3d single-valued data as well as for 2d vector-valued data (color images).

Organization. In Section 2 we introduce our general discrete mathematical model. Section 3 deals with the optimization of the codebook by the FCM algorithm. The label optimization based on the ADMM is presented in Section 4. In particular, we propose in Subsection 4.2 how to incorporate the knowledge about the layers into the data term of the functional. Synthetic numerical examples demonstrate the influence of the additional layer penalizing term. Finally, Section 5 presents real-world numerical examples, namely for the segmentation of 3d microstructural C/SiC-ceramics data obtained by micro-computed tomography and of 2d color images of traffic signs. The appendix generalizes some results to the d-dimensional setting, $d \geq 3$.

2. Mathematical Model

We start by describing our discrete mathematical model. For simplicity of notation, we restrict our attention to two-dimensional, single-valued (gray-valued) images and postpone the general d-dimensional approach to the appendix. The simple modifications which are required for vector-valued images (colored images) are addressed in the numerical part. We want to present a discrete model which uses the handy vector-matrix notation. To this end, we consider images $F : \{1, \dots, n_1\} \times \{1, \dots, n_2\} \to \mathbb{R}$ columnwise reshaped as

$$f := \mathrm{vec}_{n_1,n_2}(F) = \mathrm{vec}(F) \in \mathbb{R}^N, \quad N := n_1 n_2.$$

Let $c \in \mathbb{N}$, $c < N$ denote the number of desired segments labeled by $\mathcal{C} := \{1, \dots, c\}$. For a given image $f \in \mathbb{R}^N$ and fixed $c \in \mathbb{N}$, we are looking for *segment prototypes* in a *codebook* $r := (r_1, ..., r_c)$ and a *labeling vector* $l \in \mathcal{C}^N$ such that

$$q_{r,l} := (q_{r,l}(j))_{j=1}^N \quad \text{with} \quad q_{r,l}(j) := r_k \ \text{ if } \ l(j) = k$$

is a 'good' approximation of f. A general variational approach aims to find r and l by computing a (local) minimizer $(\hat{r}, \hat{l})$ of some functional

$$E(r,l) := \underbrace{\Phi(q_{r,l}, f)}_{\text{data term}} + \lambda \underbrace{\Psi(l)}_{\text{regularizer}}, \quad \lambda \geq 0. \tag{1}$$

Prominent examples of data terms $\Phi(q_{r,l}, f)$ are powered ℓ_p-norms

$$\Phi(q_{r,l}, f) := \sum_{j=1}^{N} |f(j) - q_{r,l}(j)|^p = \sum_{k=1}^{c} \sum_{\{j : l(j)=k\}} |f(j) - r_k|^p, \quad p \in [1, \infty), \tag{2}$$

their weighted version or the Kullback-Leibler divergence, see [6]. As regularization term $\Psi(l)$ a discrete version of the Rudin-Osher-Fatemi *TV-functional* [26] is a good candidate since it enforces 'smooth' boundaries between the labeled regions. Let us introduce such a discrete TV-functional. For a function $\mathcal{F} : \Omega \to \mathbb{R}$ from $W_{1,1}(\Omega)$, $\Omega \subset \mathbb{R}^2$, the continuous TV-functional is defined by

$$TV(\mathcal{F}) := \int_\Omega |\nabla \mathcal{F}(\mathbf{x})| \, \mathrm{d}\mathbf{x}$$

with

$$\nabla \mathcal{F}(\mathbf{x}) := (\mathcal{F}_x(\mathbf{x}), \mathcal{F}_y(\mathbf{x}))^{\mathrm{T}}, \ |\nabla \mathcal{F}(\mathbf{x})| = \left(\mathcal{F}_x^2(\mathbf{x}) + \mathcal{F}_y^2(\mathbf{x})\right)^{\frac{1}{2}}.$$

In the discrete setting, we replace the partial derivatives of $\mathcal{F}$ by forward differences of our image F. More precisely, we replace $\mathcal{F}_x$ by $D_{n_1} F$ and $\mathcal{F}_y$ by $F D_{n_2}^{\mathrm{T}}$, where

$$D_n := \begin{pmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \\ 0 & & \dots & & 0 \end{pmatrix} \in \mathbb{R}^{n \times n}$$

denotes the *forward difference matrix* and the zero row takes the mirrored (Neumann) boundary conditions into account. Since we want to deal with columnwise reshaped images we have to

introduce the tensor (Kronecker) product of matrices. The *tensor product* of $A \in \mathbb{R}^{n_1 \times n_2}$ and $B \in \mathbb{R}^{m_1 \times m_2}$ is given by the matrix

$$A \otimes B := \begin{pmatrix} a_{11}B & \dots & a_{1n_2}B \\ \vdots & \dots & \vdots \\ a_{n_1 1}B & \dots & a_{n_1 n_2}B \end{pmatrix} \in \mathbb{R}^{m_1 n_1 \times m_2 n_2}.$$

The tensor product is associative and distributive, but not commutative. Using the reshaping operator vec again, the relation

$$\operatorname{vec}(AXB^{\mathrm{T}}) = (B \otimes A)\operatorname{vec}(X) \tag{3}$$

holds true. Consequently, we obtain with the $n \times n$ identity matrix I_n that $\operatorname{vec}(D_{n_1}F) = (I_{n_2} \otimes D_{n_1})f =: f_x \in \mathbb{R}^N$ and $\operatorname{vec}(FD_{n_2}^{\mathrm{T}}) = (D_{n_2} \otimes I_{n_1})f =: f_y \in \mathbb{R}^N$, so that the *discrete gradient* of f reads in the reshaped version as

$$\begin{pmatrix} f_x \\ f_y \end{pmatrix} = \underbrace{\begin{pmatrix} I_{n_2} \otimes D_{n_1} \\ D_{n_2} \otimes I_{n_1} \end{pmatrix}}_{\nabla} f.$$

With

$$|\nabla f| := (|\nabla f|(j))_{j=1}^N, \quad |\nabla f|(j) := \left(f_x(j)^2 + f_y(j)^2\right)^{\frac{1}{2}}.$$

the discrete TV-functional becomes

$$\Psi(l) = TV(l) := \sum_{j=1}^{N} |\nabla l|(j). \tag{4}$$

Now one can try to find a minimizer of (1) by alternating the minimization of the codebook and the label vector:

Optimization of the codebook. For fixed labels l, i.e., finding

$$\operatorname*{argmin}_{r \in \mathbb{R}^c} E(r,l) = \operatorname*{argmin}_{r \in \mathbb{R}^c} \Phi(q_{r,l}, f)$$

is straightforward for various functions Φ. For example, we have for Φ in (2) with $p = 2$ that the minimizer is given by

$$r_k = \sum_{\{j : l(j) = k\}} f(j)/\#\{j : l(j) = k\}, \quad k = 1, \dots, c.$$

Optimization of the labels. Once the codebook r is known, i.e., finding

$$\operatorname*{argmin}_{l \in \mathcal{C}^N} E(r,l) = \operatorname*{argmin}_{l \in \mathcal{C}^N} \left\{\Phi(q_{r,l}, f) + \lambda TV(l)\right\},$$

is much more involved. In particular, the functional is in general not convex. For $\lambda = 0$, and Φ in (2), algorithms can be found in the literature, see [12]. For example, for $p = 2$, Lloyd's algorithm [19] can be applied and the corresponding whole alternating algorithm of codebook–label optimization is the well-known *c-means clustering algorithm*. For $\lambda \neq 0$, the optimization becomes much harder. An approach via discrete optimization for the anisotropic TV–functional which involves graph-cut based algorithms was proposed in [6].

To circumvent the difficult minimization in the label optimization, several authors have proposed to relax the problem by using instead of the label vector l a *label defining matrix* $U := (u_k(j))_{j,k=1}^{N,c}$ which assigns to each image point j a non-negative row vector $u(j)^{\mathrm{T}} := (u_k(j))_{k=1}^{c}$

with $\sum_{k=1}^{c} u_k(j) = 1$. We will see that this trick leads to a convex minimization problem in the label optimization step of an alternating algorithm. From U, the labeling vector is deduced by

$$l(j) := \underset{k=1,\dots,c}{\operatorname{argmax}}\, u_k(j). \tag{5}$$

In the following, we further use $u_k := (u_k(j))_{j=1}^N \in \mathbb{R}^N$ for the columns of U and $u := \operatorname{vec}(U) = (u_k)_{k=1}^c \in \mathbb{R}^{cN}$ for the columnwise reshaped matrix. Instead of the functional (1) with data term (2) and regularizer (4) we aim to minimize for some $p \in [1,\infty)$ and $m \geq 1$ the functional

$$\sum_{j=1}^{N}\sum_{k=1}^{c}(u_k(j))^m \underbrace{|f(j)-r_k|^p}_{s_k(j)} + \lambda \mathrm{TV}(u) \quad \text{subject to} \quad \sum_{k=1}^{c} u_k(j) = 1\,\forall j,\ u \geq 0, \tag{6}$$

where we have to define $\mathrm{TV}(u)$. The data term penalizes those $|f(j)-r_k|^p$, $k = 1,\dots,c$ for which $u_k(j)$ is large. In the ideal case that $u_k(j) = 0$ for all but one k for which it becomes 1, we resemble by (5) exactly the data term in (2). A candidate for $\mathrm{TV}(u)$ could be $\sum_{k=1}^{c} TV(u_k)$. However, in this paper we prefer a stronger coupling within u and use

$$\mathrm{TV}(u) := \| \, |(I_c \otimes \nabla)u| \, \|_1 = \sum_{j=1}^{N} |(I_c \otimes \nabla)u|(j) = \sum_{j=1}^{N} \left(|\nabla u_1(j)|^2 + \ldots + |\nabla u_c(j)|^2\right)^{\frac{1}{2}}.$$

With $s(j) := (s_k(j))_{k=1}^c$, $j = 1,\dots,N$, the functional (6) can be rewritten as

$$\sum_{j=1}^{N} \langle u(j)^m, s(j)\rangle \ + \ \lambda \| \, |(I_c \otimes \nabla)u| \, \|_1 \quad \text{subject to} \quad \sum_{k=1}^{c} u_k(j) = 1\,\forall j,\ u \geq 0. \tag{7}$$

We do not intend to find a minimizer of this non-convex functional by an alternating algorithm. Instead, we want to use this functional with $\lambda = 0$ and $m > 1$ to determine a codebook. Of course we also obtain a label defining matrix U which gives in general bad segmentation results due to the lack of regularization, but could be used as initial guess. Then, for this fixed codebook, we find a minimizer of (7) which is a convex problem now. Furthermore, we propose to incorporate the layer information into the data term. More precisely, we modify $s(j)$ to penalize u if the boundary layer is neglected.

We start with the optimization of the codebook and consider the optimization of the label vector via the matrix U afterwards.

3. Optimization of the Codebook

To find a codebook, we consider the functional (6) with $\lambda = 0$ and $m > 1$, i.e.,

$$\sum_{j=1}^{N}\sum_{k=1}^{c}(u_k(j))^m |f(j)-r_k|^p \quad \text{subject to} \quad \sum_{k=1}^{c} u_k(j) = 1\,\forall j,\ u \geq 0. \tag{8}$$

This functional is used in the *fuzzy c-means clustering algorithm* (FCM) [2], where the additional constraint $\sum_{j=1}^{N} u_k(j) > 0$ for all $k = 1,\dots,c$ appears. There exists a broad literature on the FCM algorithm and various of its generalizations, see [15] and the references therein. The notation 'fuzzy' appears from the replacement of the label vector in the ordinary c-means algorithm by the label defining matrix U.

The FCM algorithm finds a critical point (local minimizer or saddle point) of (8) by the alternating procedure described below. More precisely, the following FCM algorithm produces for an arbitrary initial vector $u^{(0)} \in \mathbb{R}^{cN}$ a sequence $(r^{(i)}, u^{(i)})$ which contains at least a subsequence that

converges to a critical point of (8), see [14]. Note that the functional (8) is non-convex in r, u, but convex with respect to r for fixed u and conversely.

The original FCM algorithm, see [3] was given for $p = 2$.

FCM Algorithm for $p = 2$:
Initialization: $u^{(0)}$
For $i = 0, \dots$ repeat until a convergence criterion is reached:

$$\begin{aligned} r_k^{(i+1)} &= \frac{\sum_{j=1}^N (u_k^{(i)}(j))^m f(j)}{\sum_{j=1}^N (u_k^{(i)}(j))^m}, \quad k = 1, \dots, c, \\ u_k^{(i+1)}(j) &= \left(\sum_{l=1}^c \left(\frac{|f(j) - r_k^{(i+1)}|^p}{|f(j) - r_l^{(i+1)}|^p} \right)^{1/(m-1)} \right)^{-1}, \quad j = 1, \dots, N, \end{aligned}$$

if $f(j) - r_k^{(i+1)} \neq 0$ for all $k = 1, \dots, c$. Supposing that the segment prototypes remain pairwise different in each iteration step, we set $u_k^{(i+1)}(j) := 1$ if $f(j) - r_k^{(i+1)} = 0$ and $u_l^{(i+1)}(j) = 0,\ l \neq k$.

Let us briefly comment on these alternating steps:

- For fixed u, the computation of a minimizer $\hat{r}$ of (8) can be handled for every k separately, so that we have to solve for every $k = 1, \dots, c$ the problem

$$\hat{r}_k := \operatorname*{argmin}_{x \in \mathbb{R}} \sum_{j=1}^N (u_k(j))^m |f(j) - x|^p.$$

  For the original FCM algorithm with $p = 2$ we obtain the above solution. For $p = 1$, the optimum $\hat{r}_k$ is the weighted median with weights $(u_k(j))^m$ of the $f(j)$, $j = 1, \dots, N$. See, e.g., [28] for weighted median computations and in connection with FCM also [16]. For $p > 1$ we refer to [20] and for $0 < p < 1$ to [22].
- For fixed r, the computation of a minimizer $\hat{u}$ of (8) can be done for every j separately, so that we have to solve for every $j = 1, \dots, N$ the problem

$$\hat{u}(j) := \operatorname*{argmin}_{x \in \mathbb{R}^c} \sum_{k=1}^c (x_k)^m \underbrace{|f(j) - r_k|^p}_{g_k} \quad \text{subject to} \quad \sum_{k=1}^c x_k = 1,\ x \geq 0. \tag{9}$$

  The *Lagrangian* of this convex problem reads

$$L(x, \lambda, \mu) = \sum_{k=1}^c \left((x_k)^m g_k - \lambda_k x_k \right) + \mu (1 - \sum_{k=1}^c x_k), \quad \lambda \geq 0$$

  and the *Karush-Kuhn-Tucker (KKT) conditions* which are necessary and sufficient for $\hat{u}(j)$ to be a minimizer are given by the constraints in (9) and

$$(\nabla_x L)_k = m (x_k)^{m-1} g_k - \lambda_k - \mu = 0 \quad \text{and} \quad x_k \lambda_k = 0, \quad k = 1, \dots, c,\ \lambda \geq 0.$$

  If there exists $k \in \{1, \dots, c\}$ with $g_k = 0$ we set $x_k := 1$ and $x_l = 0$ for $l \neq k$. Note that this is only possible for one k since the r_l are pairwise different by assumption. If $g_k \neq 0$ for all $k = 1, \dots, c$ we get by the KKT conditions that $x_k = \alpha \left(g_k\right)^{-\frac{1}{m-1}}$ with $\alpha := \left(\frac{\mu}{m}\right)^{\frac{1}{m-1}} > 0$. (Note that $x_l = 0$ for some l would imply the contradiction $\lambda_l = -\mu < 0$.) Then we conclude

by the constraints that

$$1 = \sum_{l=1}^{c} x_l = \alpha \sum_{l=1}^{c} (g_l)^{-\frac{1}{m-1}} \quad \Rightarrow \quad \alpha = 1/\sum_{l=1}^{c} (g_l)^{-\frac{1}{m-1}}$$

which results in the second formula in the FCM algorithm

$$x_k = \frac{(g_k)^{-\frac{1}{m-1}}}{\sum_{l=1}^{c} (g_l)^{-\frac{1}{m-1}}} = \left(\sum_{l=1}^{c} \left(\frac{g_k}{g_l} \right)^{\frac{1}{m-1}} \right)^{-1}.$$

4. Optimization of the Labels

Now we assume a fixed codebook r and consider the minimization of (7) for $m = 1$:

$$\sum_{j=1}^{N} \langle u(j), s(j) \rangle \; + \; \lambda \| \, |(I_c \otimes \nabla) u| \, \|_1 \quad \text{subject to} \quad \sum_{k=1}^{c} u_k(j) = 1 \, \forall j, \; u \geq 0 \tag{10}$$

with given $s(j)$. The focus on the case $m = 1$ becomes clear in Subsection 4.1, Remark 4.1. Models of this kind were considered for binary segmentation, i.e., $c = 2$, in [21]. In particular, the authors prompted to the relation with the Chan-Vese model [5]. Multi-label segmentations using the above approach or its relatives were proposed in [1, 17, 18, 23, 29]. Later, we will choose $s(j)$ in dependence on f and r, e.g., as in (6) and on the knowledge of the layer conditions. In any case, the $s(j)$, $j = 1, \ldots, N$ are fixed before the optimization. We start by describing an algorithm to solve (10) in Subsection 4.1 and consider the construction of $s(j)$ afterwards in Subsection 4.2.

4.1. Alternating Direction Method of Multipliers.

We want to solve (10) by the *alternating direction method of multipliers* (ADMM). In general, ADMM is suitable for convex optimization problems of the form

$$\underset{x \in \mathbb{R}^m, y \in \mathbb{R}^n}{\operatorname{argmin}} \; \{G_1(x) + G_2(y)\} \text{ subject to } Ax = y \tag{11}$$

with proper, closed, convex functions $G_1 : \mathbb{R}^m \to \mathbb{R} \cup \{+\infty\}$, $G_2 : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ and $A \in \mathbb{R}^{n \times m}$. The ADMM splits the problem into the following smaller subproblems which have to be solved alternatingly:

General ADMM:
Initialization: $y^{(0)}, \; b^{(0)} \in \mathbb{R}^m$ and $\gamma > 0$.
For $i = 0, \ldots$ repeat until a convergence criterion is reached:

$$\begin{aligned} x^{(i+1)} &= \underset{x \in \mathbb{R}^m}{\operatorname{argmin}} \left\{ G_1(x) + \frac{1}{2\gamma} \| b^{(i)} + Ax - y^{(i)} \|_2^2 \right\} \\ y^{(i+1)} &= \underset{y \in \mathbb{R}^n}{\operatorname{argmin}} \left\{ G_2(y) + \frac{1}{2\gamma} \| b^{(i)} + Ax^{(i+1)} - y \|_2^2 \right\} \\ b^{(i+1)} &= b^{(i)} + Ax^{(i+1)} - y^{(i+1)} \end{aligned}$$

For the above problem, the ADMM coincides with the alternating split Bregman method [13] and with the Douglas-Rachford splitting method applied to the dual problem of (11), see [8, 9, 11, 27]. For any starting values and any $\gamma > 0$, the ADMM sequences $\{b^{(i)}\}$ and $\{y^{(i)}\}$ converge to some

$\hat{b}$ and $\hat{y}$, respectively. The sequence $\{x^{(i)}\}$ converges to a solution of (11) if problem (11) has a unique solution, see, e.g., [27]. Note that $\frac{1}{\gamma}\hat{b}$ is a solution of the dual problem of (11)

$$\operatorname*{argmin}_{p\in\mathbb{R}^n}\{G_1^*(-A^*p)+G_2^*(p)\},$$

where G^* denotes the Fenchel conjugate of G.

To apply the general ADMM to our setting, we rewrite problem (10) as

$$\operatorname*{argmin}_{u,w\in\mathbb{R}^{cN},v\in\mathbb{R}^{2cN}}\{\langle u,s\rangle+\lambda\|\ |v|\ \|_1+\iota_S(w)\}\quad\text{subject to}\quad(I_c\otimes\nabla)u=v,\ I_{cN}u=w, \tag{12}$$

with the *indicator function* ι_S of S defined by

$$\iota_S(w):=\begin{cases}0 & \text{if } w\in S\\ \infty & \text{otherwise}\end{cases},\quad\text{and}\quad S:=\big\{u\in\mathbb{R}^{cN}:\sum_{k=1}^{c}u_k(j)=1\ \forall j,\ u\geq 0\big\}.$$

By definition of TV we have that $v^{\mathrm{T}}:=(v_{x,1}^{\mathrm{T}},v_{y,1}^{\mathrm{T}},\dots,v_{x,c}^{\mathrm{T}},v_{y,c}^{\mathrm{T}})$ with $v_{x,k},v_{y,k}\in\mathbb{R}^N$ and

$$|v|=\left(\sum_{k=1}^{c}(v_{x,k}^2+v_{y,k}^2)\right)^{\frac{1}{2}}\in\mathbb{R}^N. \tag{13}$$

We use $G_1(u):=\langle u,s\rangle$, $G_2(v,w):=\lambda\|\ |v|\ \|_1+\iota_S(w)$ and $A:=\begin{pmatrix}I_c\otimes\nabla\\ I_{cN}\end{pmatrix}$ in the general ADMM and obtain the following algorithm:

ADMM for (12):
Initialization: $v^{(0)}\in\mathbb{R}^{2cN},w^{(0)}\in\mathbb{R}^{cN},b^{(0)}\in\mathbb{R}^{3cN}$ and $\gamma>0$.
For $i=0,\dots$ repeat until a convergence criterion is reached:

$$u^{(i+1)}=\operatorname*{argmin}_{u\in\mathbb{R}^{cN}}\left\{\langle u,s\rangle+\frac{1}{2\gamma}\|b^{(i)}+Au-\begin{pmatrix}v^{(i)}\\ w^{(i)}\end{pmatrix}\|_2^2\right\}, \tag{14}$$

$$\begin{pmatrix}v^{(i+1)}\\ w^{(i+1)}\end{pmatrix}=\operatorname*{argmin}_{v\in\mathbb{R}^{2cN},w\in\mathbb{R}^{cN}}\left\{\lambda\|\ |v|\ \|_1+\iota_S(w)+\frac{1}{2\gamma}\|b^{(i)}+Au^{(i+1)}-\begin{pmatrix}v\\ w\end{pmatrix}\|_2^2\right\}, \tag{15}$$

$$b^{(i+1)}=b^{(i)}+Au^{(i+1)}-\begin{pmatrix}v^{(i+1)}\\ w^{(i+1)}\end{pmatrix}.$$

We have to comment the first two steps of the algorithm.

Computation of $u^{(i+1)}$. Due to the differentiability of (14) its solution follows by setting the gradient of the functional on the right-hand side to zero. Thus, the minimizer is given by the solution of the linear system of equations

$$A^{\mathrm{T}}Au=A^{\mathrm{T}}\Big(\begin{pmatrix}v^{(i)}\\ w^{(i)}\end{pmatrix}-\underbrace{\begin{pmatrix}b_v^{(i)}\\ b_w^{(i)}\end{pmatrix}}_{b^{(i)}}\Big)-\gamma s. \tag{16}$$

The ADMM works very efficient if this linear system can be solved in a fast way. This is indeed the case by the following considerations: By

$$(A\otimes B)^{\mathrm{T}}=A^{\mathrm{T}}\otimes B^{\mathrm{T}}\text{ and }(A\otimes B)(C\otimes D)=AC\otimes BD \tag{17}$$

we obtain

$$A^{\mathrm{T}}A = \left(I_c \otimes \nabla^{\mathrm{T}}\,,\, I_{cN}\right)\begin{pmatrix} I_c \otimes \nabla \\ I_{cN}\end{pmatrix} = (I_c \otimes \nabla^{\mathrm{T}}\nabla) + I_{cN}.$$

Consequently, (16) requires to solve for every $k = 1, \ldots, c$ a linear system of equations

$$(\nabla^{\mathrm{T}}\nabla + I_N)u_k = a_k \quad \text{with} \quad a_k := \nabla^{\mathrm{T}}\left(v_k^{(i)} - b_{v,k}^{(i)}\right) + \left(w_k^{(i)} - b_{w,k}^{(i)}\right) - \gamma s. \tag{18}$$

The matrix $D_n^{\mathrm{T}}D_n$ can be diagonalized by the *discrete cosine II transformation matrix* (DCT-II matrix) C_n, i.e.,

$$D_n^{\mathrm{T}}D_n = C_n^{\mathrm{T}}\,\mathrm{diag}(q_n)\,C_n, \quad q_n := \left(4\sin^2\frac{\pi j}{2n}\right)_{j=0}^{n-1} \tag{19}$$

with the orthogonal *DCT-II matrix*

$$C_n := \sqrt{\frac{2}{n}}\left(\epsilon_j \cos\frac{j(2k+1)\pi}{2n}\right)_{j,k=0}^{n-1}, \quad \epsilon_j := \begin{cases} 1/\sqrt{2} & j = 0, \\ 1 & \text{otherwise,}\end{cases} \tag{20}$$

see [24, 25]. Now straightforward computation gives

$$(\nabla^{\mathrm{T}}\nabla + I_N) = (C_{n_2}^{\mathrm{T}} \otimes C_{n_1}^{\mathrm{T}})(\Lambda + I_N)(C_{n_2} \otimes C_{n_1}),$$

where $\Lambda = I_{n_2} \otimes \mathrm{diag}(q_{n_1}) + \mathrm{diag}(q_{n_2}) \otimes I_{n_1}$, see Lemma A.1 in the appendix. Consequently, we obtain

$$u_k = (C_{n_2}^{\mathrm{T}} \otimes C_{n_1}^{\mathrm{T}})(\Lambda + I_N)^{-1}(C_{n_2} \otimes C_{n_1})a_k. \tag{21}$$

Note that the multiplication of a vector with the cosine matrix C_n can be done in a fast way with $\mathcal{O}(N \log N)$ arithmetic operations. We like to emphasize that we do not work with tensor products in our numerical computations, but with matrix - matrix operations based on (3), see (26) in the appendix.

Remark 4.1. *If $m > 1$ and $m \neq 2$ in (7), then setting the gradient of the functional in the first ADMM step to zero, leads to a non-linear system of equations. For $m = 2$ we obtain again a linear system of equations, but with coefficient matrix $2\gamma\mathrm{diag}(s) + (I_c \otimes \nabla^T\nabla) + I_{cN}$ which cannot be diagonalized by a cosine matrix. If we want to avoid the numerical solution of such a system, the so-called* primal dual hybrid gradient method *(PDHG) and its relatives [4, 10, 30] could be used as an alternative of the ADMM algorithm.*

Computation of $v^{(i+1)}$. The problem (15) can be solved separately for v and w by minimizing the following functionals (22) and (23). We start by considering

$$v^{(i+1)} = \underset{v \in \mathbb{R}^{2cN}}{\mathrm{argmin}} \left\{ \lambda \| \, |v| \, \|_1 + \frac{1}{2\gamma}\big|\big|v - \underbrace{(b_v^{(i)} + (I_c \otimes \nabla)u^{(i+1)})}_{=:g_v}\big|\big|_2^2 \right\}. \tag{22}$$

The minimizer $v^{(i+1)}$ can be obtained by the so-called *coupled shrinkage* of g_v with *threshold* $\lambda\gamma$, see [27]. Using (13) and $v(j)^{\mathrm{T}} := (v_{x,1}(j), v_{y,1}(j), \ldots, v_{x,c}(j), v_{y,c}(j))$, $j = 1, \ldots, N$, this is

$$v(j)^{(i+1)} = \begin{cases} 0 & \text{if } |g_v|(j) \leq \lambda\gamma, \\ g_v(j)\left(1 - \dfrac{\lambda\gamma}{|g_v|(j)}\right) & \text{otherwise.}\end{cases}$$

Computation of $w^{(i+1)}$. Finally we are interested in the solution of the second subproblem of (15)

$$w^{(i+1)} = \underset{w\in\mathbb{R}^{cN}}{\operatorname{argmin}} \left\{ \iota_S(w) + \frac{1}{2\gamma}\big|\big|w - \underbrace{(b_w^{(i)} + u^{(i+1)})}_{=:g_w}\big|\big|_2^2 \right\}. \tag{23}$$

In other words, we have to find the orthogonal projection of g_w onto S. This projection can also be done separately for every $j = 1, \dots, N$ which means that we have to solve N subproblems of the form

$$\underset{x\in\mathbb{R}^c}{\operatorname{argmin}} \left\{ \frac{1}{2}\|x - g\|_2^2 + \iota_{S_c} \right\} \quad S_c := \{x \in \mathbb{R}^c : \sum_{k=1}^{c} x_k = 1,\ x \geq 0\}$$

This can be done for example as in [7]. The following remark briefly explains the projection step

Remark 4.2. *The orthogonal projection of $g \in \mathbb{R}^c$ onto the hyperplane*

$$\big\{x \in \mathbb{R}^q : \sum_{k=1}^{c} x_k = \langle x, 1_c\rangle = 1\big\}$$

is given by

$$\hat{w} = g - \frac{1}{c}(\sum_{k=1}^{c} g_k - 1)1_c,$$

where 1_c is the vector with c entries 1. If $\hat{w} \geq 0$ then $\hat{w}$ is also the orthogonal projection onto S_c. If this is not the case, we set all negative components of $\hat{w}$ to zero. Let Q denote the index set of the negative components with $q := \#Q$. We project the reduced vector without zeros $(\hat{w}_k)_{k\in Q} \in \mathbb{R}^{c-q}$ onto the reduced hyperplane $\{x \in \mathbb{R}^{c-q} : \langle x, 1_{c-q}\rangle = 1\}$. We repeat the previous step until we obtain the projection onto S_c.

4.2. **Inclusion of Separating Layers.** As explained in the introduction the motivation of this paper came from a real world problem, namely the segmentation of 3d image data of Carbon/Silicon carbide (C/SiC). This image data consists of a carbon layer, a silicon layer and a silicon carbide layer. It has the characteristic property that the silicon-carbide layer separates carbon from silicon.

We call the fact that two instances are separated by a third one the *separation property*. This property will be exploited in our segmentation model.

The basic idea is to penalize the objective functional (10) by updating s such that pixels belonging to non-touching layers cannot be neighbors in our segmented image. For every label $k \in \{1, \dots, c\}$, we introduce a binary distance function

$$d_k : \{1, 2, \dots, c\} \to \{0, 1\}, \quad d_k(l) := \begin{cases} 1 & \text{if layer } l \text{ cannot touch layer } k, \\ 0 & \text{else.} \end{cases}$$

Next, we need an initial (rough) segmentation $l_0 : \{1, \dots, N\} : \to \{1, \dots, c\}$. Then we define penalizing function P_k, $k = 1, \dots, c$ as follows:

$$P_k(j) := \sum_{i\in\mathcal{N}(j)} d_k(l_0(i)), \quad j = 1, \dots, N,$$

where $\mathcal{N}(j)$ denotes the (four-star or eight-star) neighborhood of pixel j in F. Now we update s in our data term by

$$s_k(j) = |f(j) - r_k|^p + \mu P_k(j), \quad \mu > 0. \tag{24}$$

The following numerical examples demonstrate the influence of the layer penalization. In all experiments we set $\gamma := 0.01$ and $\|u^{k+1} - u^k\| < 0.1 =: \varepsilon$ as stopping criterion for ADMM. Smaller ε had no significant visual effects on the resulting images. Fig. 2(a) shows our ground truth image. It is a synthetic image consisting of three layers with gray values $r_{\text{gt}} = (0, 127, 255)$, where the subscript gt abbreviates 'ground truth'. The black and the gray part of the image are separated by a white part which is rather thin at the notches. The image was corrupted by white Gaussian noise of standard deviation 120. To obtain a codebook we smoothed the input image by convolution with a Gaussian of standard deviation $\sigma = 5$ and applied the FCM algorithm afterwards to minimize (8) with $m = p = 2$. The segmentation result is shown in Fig. 2(c). The noise in the image appears due to the lack of regularization. The corresponding codebook reads $r \approx (-1, 125, 240)$ is used to construct s in (10). Fig. 2(d) shows the result of the ADMM-segmentation. In the contrary to the FCM-result the noise disappears. But also the thin lines at the notches vanish completely.

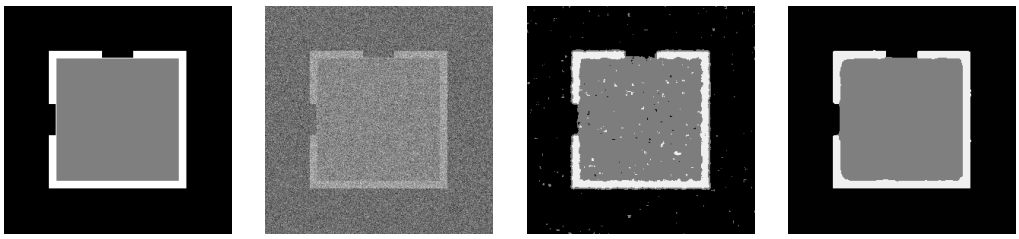
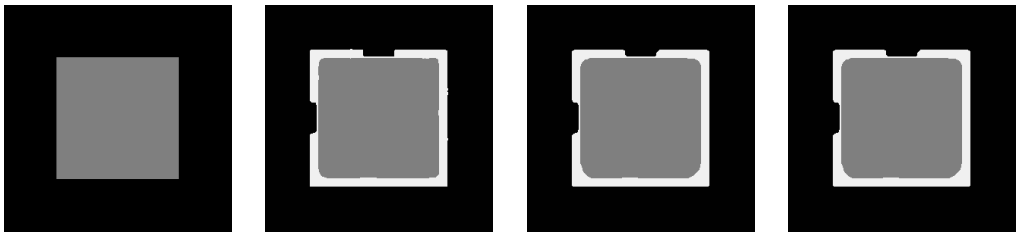


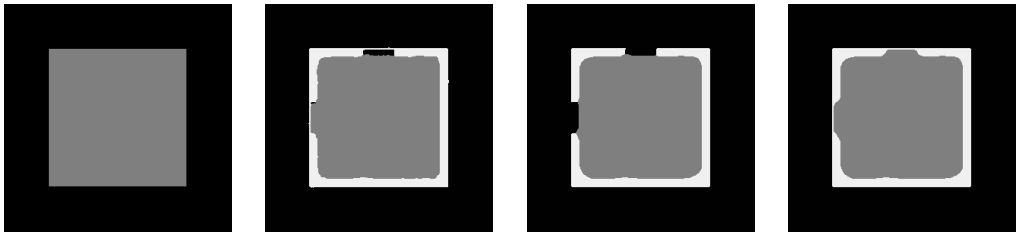
FIGURE 2. (a) Ground truth with white separating white lines which are thin at the notches. (b) Noisy image with Gaussian noise of standard deviation $\sigma = 120$, scaled to $[0, 255]$ for visualization. (c) Result of the FCM algorithm applied to the Gaussian filtered image. (d) Solution of ADMM applied to b) without layer penalization, $\lambda = 250$.



(a) initial segmentation (b) $\lambda = 250, \mu = 1500$ (c) $\lambda = 500, \mu = 500$ (d) $\lambda = 500, \mu = 1500$

FIGURE 3. Solutions with layer penalization with respect to the inner square in (a) as initial segmentation for different parameters λ and μ.

The effect of the layer penalization (24) is shown in Figs. 3 and 4. In Fig. 3 the inner square of the ground truth is used as initial segmentation. Due to the layer penalization (24) the thin lines at the notches are preserved. We demonstrate also the effects of different parameters λ and μ. In Fig. 4 the outer square of the ground truth is used as initial segmentation. The outer square as initial segmentation causes the thin white lines to vanish, because the corresponding pixels are not penalized at all. The different solutions in Figures 4(b), 4(c) and 4(d) are all valid with respect to the outer square as initial segmentation because black next to black and black next to white are



(a) initial segmentation (b) $\lambda = 250, \mu = 1500$ (c) $\lambda = 500, \mu = 500$ (d) $\lambda = 500, \mu = 1500$

FIGURE 4. Solutions with layer penalization with respect to the outer square in (a) as initial segmentation for different parameters λ and μ.

both allowed configurations. That is, on the boundary of the outer square appears no black pixel with a gray neighbor or vice versa. In all solutions the original shape of the image is preserved because the segmentation does not only depend on the penalization of the boundary but also on the original image.

Remark 4.3. *Lellmann et al. [18] have also considered a more general class of TV–based regularizers with a weights matrix M*

$$\mathrm{TV}(u) := \| \, |(M \otimes \nabla)u| \, \|_1.$$

For our applications this can be used to modify the regularization corresponding to a distance $d(k, l)$ between labels k and l of adjoined regions.

5. Numerical Experiments

In this section we demonstrate the good performance of our algorithm for real-world data. For the implementation we used MATLAB and executed our experiments on an Intel Xeon 2.5 GHz processor. For the codebook retrieval we applied the FCM algorithm introduced in Section 3 with parameters $p = m = 2$. We have found that the subsequent label-finding step is very robust with respect to the codebook.

5.1. Segmentation of Carbon/Silicon carbide (C/SiC) image data.

As already mentioned in the introduction, the C/SiC image data consists of a carbon layer, a silicon layer and a silicon carbide layer which separates the carbon layer from the silicon layer. Actually, there is a fourth layer in the C/SiC material, namely the pores in the carbon. These are the darkest areas in the image and which can be simply found by, e.g., by thresholding. Therefore we ignore them in the following. Then the carbon has the darkest gray of the three other layers and the separating silicon carbide the lightest. Accordingly the materials are represented in the segmented results. There are artefacts in form of circles visible in the input images which should not appear in the segmentation, because they are not part of the material but arise due to the image capturing method.

2d segmentation of a slice of the C/SiC image data. In our first experiment, we segmented a 2d slice of the size 1251×1251. Due to the lack of noise it was not necessary to apply a smoothing filter before computing the codebook. We used as codebook the result of the FCM algorithm $r \approx (71, 100, 120)$. The initial segmentation was the result of the ADMM algorithm with only 5 iterations and ADMM parameter $\gamma := 0.01$. Fig. 5 shows the corresponding results. The ADMM applied to the original model (even after much more iteration steps) eliminates the ringing artefacts , but cannot find the separating SiC layer everywhere as our layer penalizing model did. We have

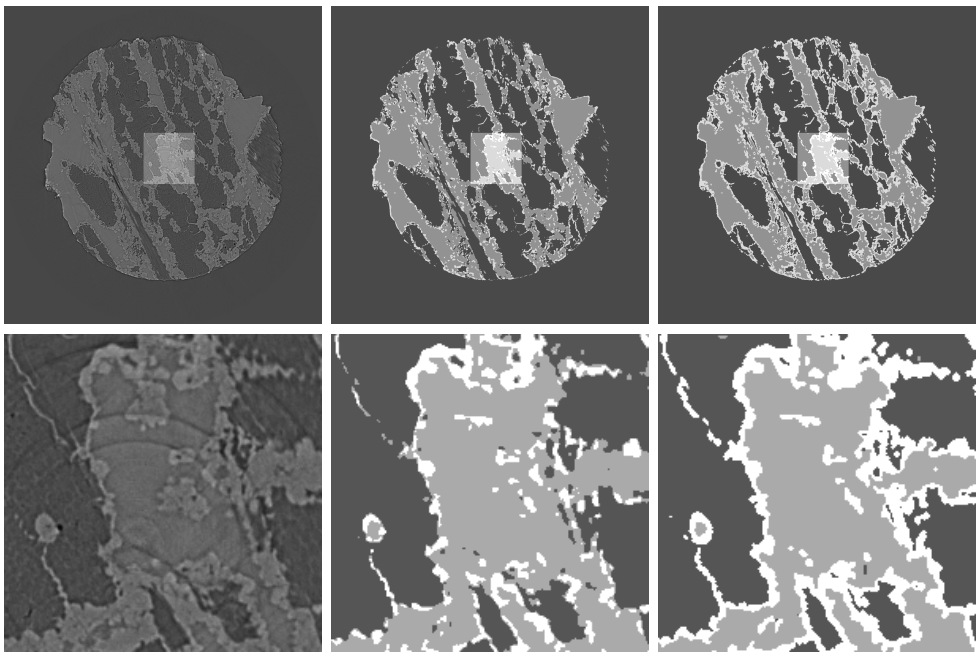


FIGURE 5. 2d segmentation of a C/SiC slice. The first row shows the full images, while the second row presents zooms into the marked areas of the images. (a) Input image with materials from light to dark: silicon carbide, silicon, carbon, pores in carbon (ignored). (b) Initial segmentation with 5 ADMM iterations without layer penalization and $\lambda = 10$. (c) Final segmentation with 40 ADMM iterations applied to our model with layer penalization and $\lambda = 10$, $\mu = 200$.

used the ADMM for the layer penalizing model with 40 iteration steps. The overall algorithm including the computation time for the FCM and the pre-segmentation requires about 630 seconds, but the code can further be optimized.

3d segmentation of C/Sic image data. Our next input image has the size of $300 \times 300 \times 100$ pixels. We used an arbitrary selection from the original image data. The discrete 3d gradient and some 3d modifications required in the first step of the ADMM algorithm are described in the appendix. To establish the codebook we applied the FCM algorithm which results in $r \approx (126, 151, 175)$. Figs. 6 and 7 show our 3d segmentation results for 4, resp., 6 arbitrarily chosen subsequent slices (slices: 10,12,14,16, resp. 10,12,14,16,18,20) in x-y, resp., x-z direction. As initial segmentation we used the ADMM results for the model without layer penalization with 5 iterations and $\gamma := 0.1$ shown in the second rows of the figures. The markers in the figures indicate that not everywhere the separation property was fulfilled. Then we exploited the layer penalized model and executed 30 ADMM iterations. The improvement can be seen in the third row of the figures. The overall computation was approximately 1900 seconds. Increasing the iterations of the initial segmentation did not change our final results significantly.

5.2. **2d segmentation of colored images.** Finally, we show that our segmentation model is also suitable for colored images. We used RGB colorization. In this case we assign to every image pixel j a color vector $f(j) \in \mathbb{R}^3$ and for every segment k a color prototype $r_k \in \mathbb{R}^3$. Then we have only to replace $|r_k - f(j)|$ by the vector norm $s_k(j) = \|r_k - f(j)\|_2$. We applied our method to a 3-class segmentation of German traffic signs taken from "The German Traffic Sign Recognition

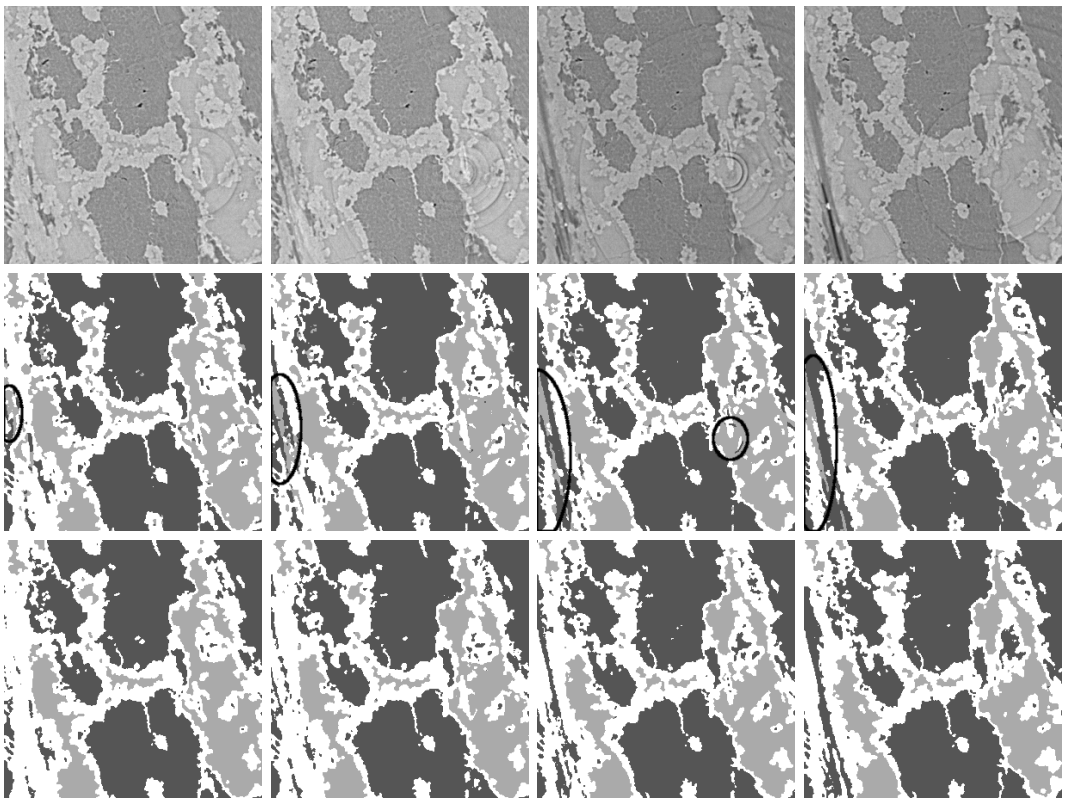


FIGURE 6. First row: x-y-slices of 3d input image data Second row: 3d initial segmentation with 5 ADMM iterations without layer penalization and $\lambda = 15$. Some faulty segmented regions are marked for visualization. Third row: Final segmentation with 30 ADMM iterations applied to our model with layer penalization and $\lambda = 15$, $\mu = 500$.

Benchmark" of the Ruhr-Universität Bochum. These traffic signs have white boundaries separating the red area from the background. The results are shown in Fig. 8. To compute the codebook we applied the FCM algorithm. For the first and second row, resp., of Fig. 8 we got

$$r_1 \approx \begin{pmatrix} 187 & 251 & 74 \\ 67 & 241 & 64 \\ 58 & 243 & 52 \end{pmatrix}, \quad r_2 \approx \begin{pmatrix} 137 & 37 & 109 \\ 136 & 37 & 64 \\ 139 & 34 & 55 \end{pmatrix},$$

where every column vector is the prototype of a segment. We used ADMM without layer penalization to generate the initial segmentation with 10 iterations and 30 iterations for the final segmentation step with layer penalization. The ADMM parameter was $\gamma := 0.05$. The improvement is clearly visible. As in the previous examples more iterations did not improve our results significantly.

Acknowledgment. The authors like to thank Jürgen Meinhardt, Fraunhofer ISC Würzburg, and Alexander Rack, ESFR Grenoble, for providing us with the C/SiC image data.

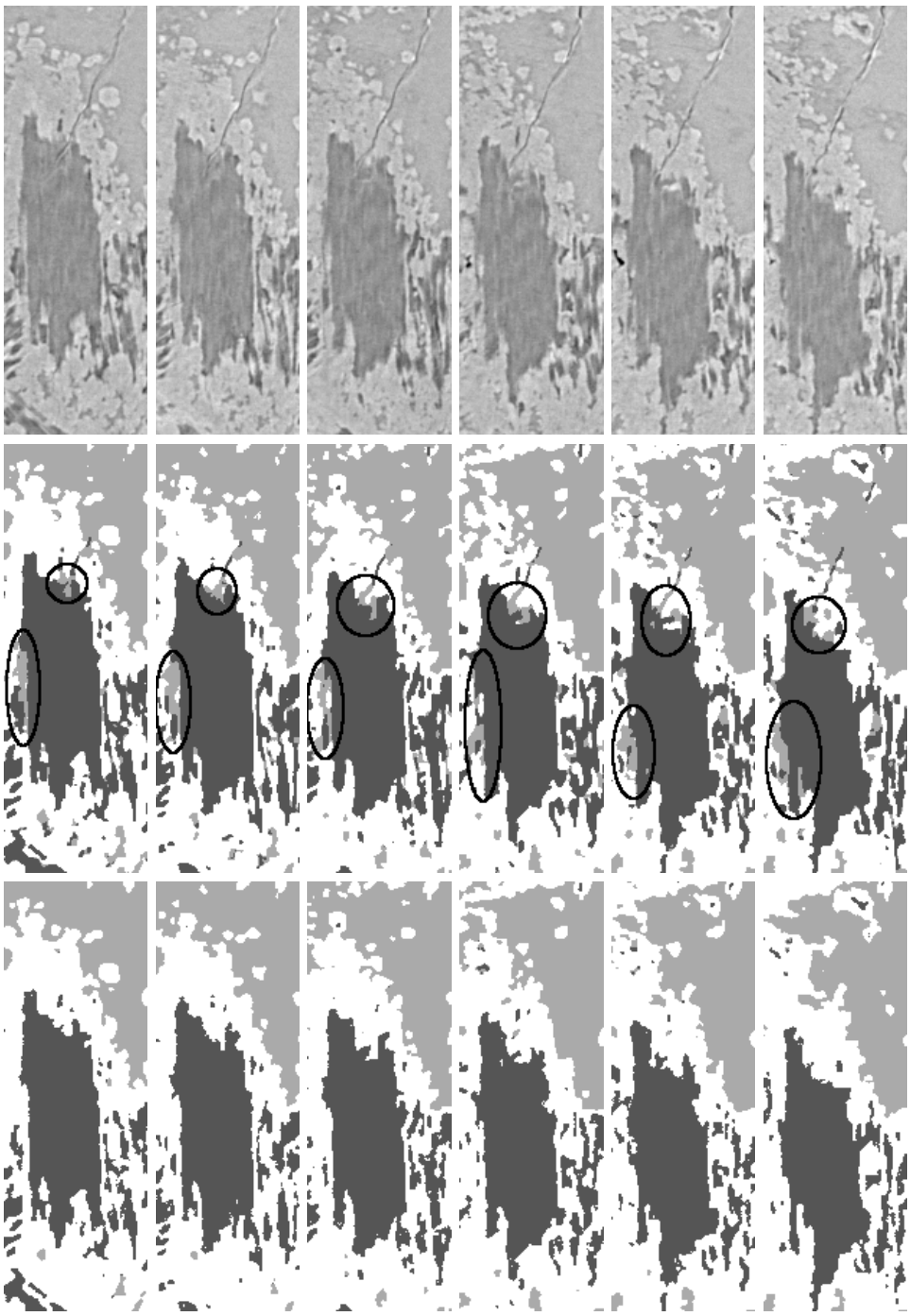


FIGURE 7. x-z-slices of our 3d segmentation results as in Fig. 6.

APPENDIX A. ASPECTS OF d-DIMENSIONAL IMAGE SEGMENTATION

Finally, we consider the d-dimensional setting, $d \geq 3$. More precisely, while most generalizations to higher dimensions in the algorithms are straightforward, the efficient solution of the linear system

FIGURE 8. Segmentation of two different traffic signs . (a) Input images. (b) Initial segmentation without layer penalization after 10 ADMM iterations. (c) Final segmentation with layer penalization after 40 ADMM iterations. In the first row we used $\lambda = 100, \mu = 500$ and in the second row $\lambda = 50, \mu = 300$.

of equations (18) in the first ADMM step requires some explanation. To this end, we consider d-dimensional images $F : \{1, \dots, n_1\} \times \cdots \times \{1, \dots, n_d\} \to \mathbb{R}$ and reshape them indexwise starting with the first index to get a vector $f \in \mathbb{R}^N$ with $N := \prod_{i=1}^{d} n_i$. We introduce the *d-dimensional discrete gradient* by

$$\nabla := \begin{pmatrix} \nabla_1 \\ \nabla_2 \\ \vdots \\ \nabla_d \end{pmatrix} \quad \text{with } \nabla_i := \bigotimes_{j=d}^{i+1} I_{n_j} \otimes D_{n_i} \otimes \bigotimes_{j=i-1}^{1} I_{n_j} = I_{\beta_i} \otimes D_{n_i} \otimes I_{\alpha_i},$$

where $\alpha_i := \prod_{j=1}^{i-1} n_j$, $\beta_i := \prod_{j=i+1}^{d} n_j$ and

$$\bigotimes_{i=d}^{1} M_i := M_d \otimes M_{d-1} \otimes \cdots \otimes M_2 \otimes M_1.$$

To solve the d-dimensional version of the linear system of equations (18), we use the following lemma.

Lemma A.1. *The negative discrete d-dimensional Laplacian $\nabla^T\nabla$ can be diagonalized by tensor products of DCT-II matrices C_n defined in* (20) *as follows:*

$$\nabla^T\nabla = \left(\bigotimes_{j=d}^{1} C_{n_j}^T\right) \Lambda \left(\bigotimes_{j=d}^{1} C_{n_j}\right),$$

where

$$\Lambda := \sum_{i=1}^{d} \left(I_{\beta_i} \otimes \ \mathrm{diag}(q_{n_i}) \ \otimes I_{\alpha_i}\right), \quad q_n := \left(4\sin^2\frac{\pi j}{2n}\right)_{j=0}^{n-1}.$$

Proof. Using (17) and (19), we obtain

$$\nabla^T\nabla = \left(\nabla_1^T\nabla_1 + \ldots \ + \nabla_d^T\nabla_d\right) = \sum_{i=1}^{d} \left(I_{\beta_i} \otimes D_{n_i}^{\mathrm{T}} D_{n_i} \otimes I_{\alpha_i}\right) = \sum_{i=1}^{d} \left(I_{\beta_j} \otimes C_{n_i}^{\mathrm{T}} \ \mathrm{diag}(q_{n_i}) \ C_{n_i} \otimes I_{\alpha_j}\right).$$

By the orthogonality of the cosine matrix $I_{n_i} = C_{n_i}^{\mathrm{T}} C_{n_i}$ and (17) we get further

$$\begin{aligned}
\nabla^{\mathrm{T}}\nabla &= \sum_{i=1}^{d} \Big(\bigotimes_{j=d}^{i+1} C_{n_j}^{\mathrm{T}} C_{n_j}\Big) \otimes \left(C_{n_i}^{\mathrm{T}} \,\mathrm{diag}(q_{n_i}) C_{n_i}\right) \otimes \Big(\bigotimes_{j=i-1}^{1} C_{n_j}^{\mathrm{T}} C_{n_j}\Big) \\
&= \sum_{i=1}^{d} \underbrace{\left(C_{n_d}^{\mathrm{T}} \otimes C_{n_{d-1}}^{\mathrm{T}}\right)\left(C_{n_d} \otimes C_{n_{d-1}}\right)}_{C_{n_d}^{\mathrm{T}} C_{n_d} \otimes C_{n_{d-1}}^{\mathrm{T}} C_{n_{d-1}}} \otimes \Big(\bigotimes_{j=d-2}^{i+1} C_{n_j}^{\mathrm{T}} C_{n_j}\Big) \otimes \left((C_{n_i})^{\mathrm{T}} \ \mathrm{diag}(q_{n_i}) \ C_{n_i}\right) \otimes \Big(\bigotimes_{j=i-1}^{1} C_{n_j}^{\mathrm{T}} C_{n_j}\Big).
\end{aligned}$$

Applying the previous step also to the last term and repeating it iteratively we obtain

$$\nabla^{\mathrm{T}}\nabla = \sum_{i=1}^{d} \Big(\bigotimes_{j=d}^{i+1} C_{n_j}^{\mathrm{T}}\Big)\Big(\bigotimes_{j=d}^{i+1} C_{n_j}\Big) \ \otimes \ \left(C_{n_i}^{\mathrm{T}} \,\mathrm{diag}(q_{n_i}) \ C_{n_i}\right) \ \otimes \Big(\bigotimes_{j=i-1}^{1} C_{n_j}^{\mathrm{T}}\Big)\Big(\bigotimes_{j=i-1}^{1} C_{n_j}\Big).$$

Using (17) results in

$$\begin{aligned}
\nabla^{\mathrm{T}}\nabla &= \sum_{i=1}^{d} \underbrace{\Big(\bigotimes_{j=d}^{i} C_{n_j}^{\mathrm{T}}\Big)}_{A} \underbrace{\Big(\Big(\bigotimes_{j=d}^{i+1} C_{n_j}\Big) \otimes \ \mathrm{diag}(q_{n_i}) \ C_{n_i}\Big)}_{B} \otimes \underbrace{\Big(\bigotimes_{j=i-1}^{1} C_{n_j}^{\mathrm{T}}\Big)}_{C} \underbrace{\Big(\bigotimes_{j=i-1}^{1} C_{n_j}\Big)}_{D} \\
&= \sum_{i=1}^{d} \Big(\bigotimes_{j=d}^{1} C_{n_j}^{\mathrm{T}}\Big) \left(\Big(\bigotimes_{j=d}^{i+1} C_{n_j}\Big) \otimes \ \mathrm{diag}(q_{n_i}) \ C_{n_i} \otimes \Big(\bigotimes_{j=i-1}^{1} C_{n_j}\Big)\right) \\
&= \sum_{i=1}^{d} \Big(\bigotimes_{j=d}^{1} C_{n_j}^{\mathrm{T}}\Big) \left(I_{\beta_i}\Big(\bigotimes_{j=d}^{i+1} C_{n_j}\Big) \otimes \ \mathrm{diag}(q_{n_i}) \ C_{n_i} \otimes I_{\alpha_i}\Big(\bigotimes_{j=i-1}^{1} C_{n_j}\Big)\right) \\
&= \sum_{i=1}^{d} \Big(\bigotimes_{j=d}^{1} C_{n_j}^{\mathrm{T}}\Big) \left(I_{\beta_i} \otimes \ \mathrm{diag}(q_{n_i}) \ \otimes I_{\alpha_i}\right) \Big(\bigotimes_{j=d}^{1} C_{n_j}\Big).
\end{aligned}$$

□

By the lemma, the d-dimensional version of (21) is given by

$$u_k = \left[\left(\bigotimes_{j=d}^{1} C_{n_j}^{\mathrm{T}} \right) (\Lambda + I_N)^{-1} \left(\bigotimes_{j=d}^{1} C_{n_j} \right) \right] a_k. \tag{25}$$

Tensor products of matrices are useful to write the models in a matrix-vector form. However, in our numerical computations we will not work with tensor products. In the following, we describe how this can be avoided. We start with the case $d = 2$. For 2-dimensional images we consider

$$(\nabla^{\mathrm{T}}\nabla + I_N)^{-1} f = \left(C_{n_2}^{\mathrm{T}} \otimes C_{n_1}^{\mathrm{T}}\right) (\Lambda + I_N)^{-1} \left(C_{n_2} \otimes C_{n_1}\right) f.$$

By (3) we have

$$\left(C_{n_2} \otimes C_{n_1}\right) f = \mathrm{vec}\left(C_{n_1} F\, C_{n_2}^{\mathrm{T}}\right).$$

Denoting the componentwise multiplication (Hadamard product) of two matrices by $\circ$ we obtain

$$(\Lambda + I_N)^{-1} \mathrm{vec}\left(C_{n_1} F\, C_{n_2}^{\mathrm{T}}\right) = \mathrm{vec}\left(L \circ \left(C_{n_1} F\, C_{n_2}^{\mathrm{T}}\right)\right)$$

where $L \in \mathbb{R}^{n_1 \times n_2}$ is the columnwise reshaping of the diagonal of $(\Lambda + I_N)^{-1}$ into a matrix. Applying (3) again leads to

$$(\nabla^{\mathrm{T}}\nabla + I_N)^{-1} f = \left(C_{n_2}^{\mathrm{T}} \otimes C_{n_1}^{\mathrm{T}}\right) \mathrm{vec}\left(L \circ \left(C_{n_1} F\, C_{n_2}^{\mathrm{T}}\right)\right) = \mathrm{vec}\left(C_{n_1}^{\mathrm{T}} L \circ \left(C_{n_1} \cdot F \cdot C_{n_2}^{\mathrm{T}}\right) C_{n_2}\right). \tag{26}$$

For the general d-dimensional case we have implemented the right-hand side of (25) without constructing the tensor products explicitly by applying the following lemma.

Lemma A.2. *Let $f \in \mathbb{R}^N$ with $N = n_1 \dots n_d$ be given. We set $N_k := n_1 \dots n_k$, $k = 1, \dots, d$. Starting with $f^{k_d} := f$, $k_d = 1$, we define successively:*

- $F^{k_d} := \mathrm{vec}^{-1}_{N_{d-1}, n_d}(f^{k_d}) \in \mathbb{R}^{N_{d-1} \times n_d}$,
- $f^{k_d, k_{d-1}}$ *is the* k_{d-1}*-th column of* F^{k_d}, $k_{d-1} = 1, \dots, n_d$,
- $F^{k_d, k_{d-1}} := \mathrm{vec}^{-1}_{N_{d-2}, n_{d-1}}(f^{k_d, k_{d-1}}) \in \mathbb{R}^{N_{d-2} \times n_{d-1}}$
- *and so on.*

Then, for $j = 2, \dots, d$ and $A_i \in \mathbb{R}^{n_i \times n_i}$ the following relation holds true:

$$\left(\bigotimes_{i=j}^{1} A_i \right) f^{k_d, \dots, k_j} = \mathrm{vec}_{N_{j-1}, n_j} \left(\left(\bigotimes_{i=j-1}^{1} A_i \right) F^{k_d, \dots, k_j} A_j^T \right).$$

Proof. Apply (3) to every column k_j for every j. □

References

[1] E. Bae, J. Yuan, and X.-C. Tai. Global minimization for continuous multiphase partitioning problems using a dual approach. CAM Report 09-75, UCLA, 2009.

[2] J. C. Bezdek. A convergence theorem for the fuzzy isodata clustering algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(1):1 –8, 1980.

[3] J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Plenum Press, New York, 1981.

[4] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. Preprint, University of Graz, 2010.

[5] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE Transactions on Image Processing*, 10(2):266–277, 2001.

[6] C. Chaux, A. Jezierska, J.-C. Pesquet, and H. Talbot. A spatial regularization approach for vector quantization. *Journal of Mathematical Imaging and Vision*, to appear.
[7] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l1-ball for learning in high dimensions. In *ICML '08 Proceedings of the 25th International Conference on Machine Learning*, ACM New York, 2008.
[8] J. Eckstein and D. P. Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming*, 55:293–318, 1992.
[9] E. Esser. Applications of Lagrangian based alternating direction methods and connections to split Bregman. CAM Report 09-31, UCLA, 2009.
[10] E. Esser, X. Zhang, and T. F. Chan. A general framework for a class of first order primal-dual algorithms for TV minimization. CAM Report 09-67, UCLA, 2009.
[11] D. Gabay. Applications of the method of multipliers to variational inequalities. In M. Fortin and R. Glowinski, editors, *Augmented Lagrangian Methods: Applications to the Solution of Boundary Value Problems*, chapter IX, pages 299–340. North-Holland, Amsterdam, 1983.
[12] A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression.* Kluwer Academic Press, MA, USA, 1992.
[13] T. Goldstein and S. Osher. The split Bregman method for l1-regularized problems. *SIAM Journal on Imaging Sciences*, 2(2):323–343, 2009.
[14] R. J. Hathaway and J. C. Bezdek. Recent convergence results for the fuzzy c-means clustering algorithms. *Journal of Classification*, 5:237–247, 1988.
[15] R. J. Hathaway, J. C. Bezdek, and Y. Hu. Generalized fuzzy c-means clustering strategies using L_p norm distances. *IEEE Transactions on Fuzzy Systems*, 8:576–582, 2000.
[16] P. R. Kersten. Fuzzy order statistics and their application to fuzzy clustering. *IEEE Transactions on Fuzzy Systems*, 7:708–712, 1999.
[17] J. Lellmann, J. Kappes, F. Becker, and C. Schnörr. Convex multi-class image labeling with simplex-constrained total variation. In X.-C. Tai, K. Morken, M. Lysaker, and K.-A. Lie, editors, *Scale Space and Variational Methods, volume 5567 of LNCS*, volume 5567 of *Lecture Notes in Computer Science*, pages 150–162. Springer, 2009.
[18] J. Lellmann and C. Schnörr. Continuous multiclass labeling approaches and algorithms. *Preprint, University of Heidelberg*, 2010.
[19] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, Mar. 1982.
[20] S. Miyamoto and Y. Agusta. Efficient algorithms for l_p fuzzy c-means and their termination properties. In *Proc. 5th Conference of the International Federation Classification Socienty*, pages 255–258, Kobe, Japan, 1996.
[21] M. Nikolova, S. Esedoglu, and T. F. Chan. Algorithms for finding global minimizers of image segmentation and denoising models. *SIAM Journal on Applied Mathematics*, 66(5):1632–1648, 2006.
[22] D. D. Overstreet. Generalized fuzzy c-means clustering. Master's thesis, Georgia Southern University, 1998.
[23] T. Pock, A. Chambolle, D. Cremers, and H. Bischof. A convex relaxation approach for computing minimal partitions. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 810–817, 2009.
[24] D. Potts and G. Steidl. Optimal trigonometric preconditioners for nonsymmetric toeplitz systems. *Linear Algebra and its Applications*, 281:265–292, 1998.
[25] K. R. Rao and P. Yip. *Discrete Cosine Transform.* Academic Press, New York, 1990.

[26] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259–268, 1992.
[27] S. Setzer. Operator splittings, Bregman methods and frame shrinkage in image processing. *International Journal of Computer Vision*, 2010. accepted.
[28] S. Setzer, G. Steidl, and T. Teuber. On vector and matrix median computation. *Journal of Computational and Applied Mathematics*, 2011. accepted.
[29] C. Zach, D. Gallup, J.-M.Frahm, and M. Niethammer. Fast global labeling for real-time stereo using multiple plane sweeps. *Vision, Modeling, and Visualization Workshop*, 2008.
[30] M. Zhu and T. F. Chan. An efficient primal-dual hybrid gradient algorithm for total vatiation image restoration. *UCLA CAM Report 08-34*, 2008.

Fraunhofer ITWM, Fraunhofer Platz 1, 67663 Kaiserslautern, Germany

University of Kaiserslautern, Department of Mathematics, Gottlieb Daimler Str., 67663 Kaiserslautern, Germany