TIGHT CONVEX RELAXATIONS FOR VECTOR-VALUED LABELING

BASTIAN GOLDLUECKE[1], EVGENY STREKALOVSKIY[2] AND DANIEL CREMERS[2]
[1]HEIDELBERG COLLABORATORY FOR IMAGE PROCESSING [2]TU MUENCHEN

Abstract. Multi-label problems are of fundamental importance in computer vision and image analysis. Yet, finding global minima of the associated energies is typically a hard computational challenge. Recently, progress has been made by reverting to spatially continuous formulations of respective problems and solving the arising convex relaxation globally. In practice this leads to solutions which are either optimal or within an a posteriori bound of the optimum. Unfortunately, in previous methods, both run time and memory requirements scale linearly in the total number of labels, making them very inefficient and often not applicable to problems with higher dimensional label spaces.

In this paper, we propose a reduction technique for the case that the label space is a continuous product space and the regularizer is separable, i.e. a sum of regularizers for each dimension of the label space. On typical real-world labeling problems, the resulting convex relaxation requires orders of magnitude less memory and computation time than previous methods. This enables us to apply it to large-scale problems like optic flow, stereo with occlusion detection, segmentation into a very large number of regions, and joint denoising and local noise estimation. Experiments show that despite the drastic gain in performance, we do not arrive at less accurate solutions than the original relaxation. Using the novel method, we can for the first time efficiently compute solutions to the optic flow functional which are within provable bounds (typically 5%) of the global optimum.

Key words. Multi-label problems, algorithms, duality, convex relaxation.

AMS subject classifications. 68U10, 49M29, 65K10

1. Introduction.

1.1. The Multi-labeling Problem. Recently, there has been a surge of research activity on convex relaxation techniques for energy minimization in computer vision. Particular efforts were directed towards binary and multilabel problems, as they lie at the heart of fundamental problems like segmentation [23, 21, 9, 41], stereo [27], 3D reconstruction [11], Mumford-Shah denoising [26] and optic flow [14].

The aim is to assign to each point x of a domain $\Omega \subset \mathbb{R}^n$ a *label* from a set $\Gamma \subset \mathbb{R}^d$. Assigning the label $\gamma \in \Gamma$ to x is associated with the *cost* $c^\gamma(x) = c(x, \gamma) \in \mathbb{R}$. In computer vision applications, this local cost denotes how well a given labeling fits some observed data. They can be arbitrarily sophisticated, derived from statistical models or complicated local matching scores. In the following, we will assume that the cost functions c^γ lie in the Hilbert space of square integrable functions $\mathcal{L}^2(\Omega)$. Aside from minimizing the local costs, we want the optimal assignment to exhibit a certain regularity. We enforce this requirement by penalizing each possible labeling $\boldsymbol{u} : \Omega \to \Gamma$ with a *regularizer* $J(\boldsymbol{u}) \in \mathbb{R}$. This regularizer reflects our knowledge about which label configurations are a priori more likely, and typically enforces a form of spatial coherence of the computed labeling.

Finding a labeling $\boldsymbol{u} : \Omega \to \Gamma$ which minimizes the sum of data term and regularizer, i.e.

$$\operatorname*{argmin}_{\boldsymbol{u} \in \mathcal{L}^2(\Omega,\Gamma)} \left\{ J(\boldsymbol{u}) + \int_\Omega c(x, \boldsymbol{u}(x)) \, \mathrm{d}x \right\} \tag{1.1}$$

is a hard computational challenge as the overall energy is generally not convex. For some cases, good results may be obtained by local minimization, starting from a

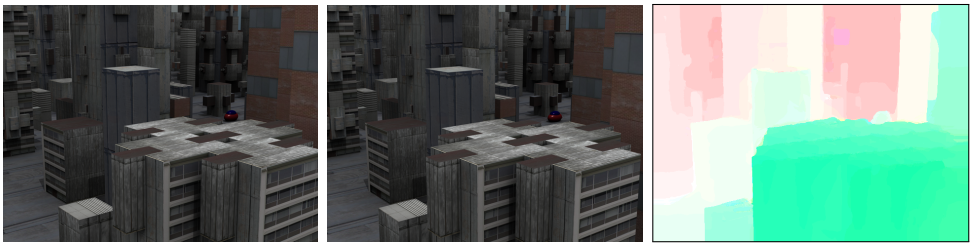


Fig. 1.1: *The proposed relaxation method can approximate the solution to multi-labeling problems with a huge number of possible labels by globally solving a convex relaxation model. This example shows two images and the optic flow field between them, where flow vectors were assigned from a possible set of* 50×50 *vectors, with truncated linear distance as a regularizer. The problem has so many different labels that a solution cannot be computed by alternative relaxation methods on current hardware.*

good initialization, possibly further improved by coarse-to-fine strategies commonly employed in optical flow estimation. Yet, such methods cannot guarantee any form of quality of the result and performance typically depends on data, on initialization and on the choice of algorithmic minimization scheme (number of levels in the coarse-to-fine hierarchy, number of iterations per level, etc.). The goal of this paper is to develop solutions to such problems which do not depend on initialization and which lie within a computable bound of the global optimum.

1.2. Contribution: Product label spaces. In this work, we consider label spaces which can be written as a product of a finite number d of spaces, $\Gamma = \Lambda_1 \times \cdots \times \Lambda_d$. The central idea is as follows. Assume that the spaces Λ_k are discrete or have been discretized, and let N_k be the number of elements in Λ_k. Then the total number of labels is $N = N_1 \cdot \ldots \cdot N_d$. In previous relaxations for the multi-label problem, this means that we need to optimize over a number of N binary indicator functions, which can easily amount to thousands of indicator functions in practical problems. In order to make problems of this form feasible to solve, we present a reduction method which only requires $N_1 + \cdots + N_d$ binary functions. As a consequence, memory grows linearly (rather than exponentially) in the number of dimensions, while computation time is greatly reduced.

An important limitation, however, is that we only consider *separable* regularizers of the form

$$J(\boldsymbol{u}) = \sum_{k=1}^{d} J_k(u_k), \tag{1.2}$$

which means that J acts on the label components u_k of $\boldsymbol{u}$ independently.

We will show that with the novel reduction technique, it is possible to efficiently solve convex relaxations to multi-label problems which are far too large to approach with existing techniques. A prototypical example is optic flow, where the total number of labels is typically around 32^2 for practical problems. In that case, for example, we only require 64 indicator functions instead of 1024. However, the proposed method applies to a much larger class of labeling problems. This reduction in variable size not only allows for substantially higher resolution of the label space, but it also gives rise to a drastic speedup.

The present paper subsumes and extends two previous conference publications [14, 36]. For this journal paper, both earlier works were integrated into a comprehensive exposition of continuous vectorial labeling problems. Contents of the earlier paper [14] were revised in light of the new findings and improved relaxations in [36], and the differences between both approaches illuminated. Compared to the original conference publications, we provide a more detailed background and more complete theory. We included the total cyclic variation [35] as an additional regularizer and extended the description of the algorithm we use to implement the method. In particular, we were able to prove that our new data term relaxation corresponds to the exact convex envelope for the non-convex data term of the energy, which makes it the tightest possible relaxation for the data term. All experiments in the older paper were redone and updated with respect to the novel implementation of the algorithm in [36]. We also included a few more more experimental comparisons, in particular to discrete methods, and extended the implementation to arbitrary label space dimension. Complete source code to reproduce the experiments is publicly available on Sourceforge under a GPL3 license as part of our CUDA library for continuous convex optimization in image processing [1].

2. Related work.

2.1. Discrete approaches. It is well known that in the fully discrete setting, the minimization problem (1.1) is equivalent to maximizing a Bayesian posterior probability, where the prior probability gives rise to the regularizer [37]. The problem can be stated in the framework of Markov random fields (MRF) and discretized using a graph representation, where the nodes denote discrete pixel locations and the edge weights encode the energy functional [4].

Fast combinatorial minimization methods based on graph cuts can then be employed to search for a minimizer. In the case that the label space is binary and the regularizer submodular, a global solution of (1.1) can be found by computing a minimum cut [15, 20]. Continuous variants of this minimum cut problem have also been studied [34]. For multi-label problems, one can approximate a solution for example by solving a sequence of binary problems (α-expansions) [6, 31], linear programming relaxations [40] or quadratic pseudo-boolean optimization [19]. Exact solutions to multi-label problems can only be found in some special cases. An important case are multi-label problems over a linearly ordered label set with convex regularizer. The global optimum of this problem corresponds to a cut in a multi-layered graph which can be computed in polynomial time [16]. A different discrete encoding scheme for this problem was also presented in [31].

In [32, 33] the problem of image registration is formulated as an MRF labeling problem, which is minimized via LP relaxation. The authors present a decoupling strategy for the displacement components which is related to ours, albeit only applicable in the discrete case. It allows a simplification of the graph and consequently larger numbers of labels. Another discrete method which is related to our work is [29]. The authors present a compact encoding scheme for the multilabel problem called a log-transformation which makes the unary term non-submodular. This is in analogy to our transformation, which makes the previously convex data term non-convex. The problem of large label spaces is also tackled in [13], where the authors compute optical flow from an MRF labeling problem using a lower dimensional parametric description for the displacements.

[1] `https://sourceforge.net/p/cocolib`

However, in many important scenarios the label space cannot be ordered. Moreover, a non-convex regularizer is often more desirable to better preserve discontinuities in the solution. Even for relatively simple non-convex regularizers like the Potts distance, the resulting combinatorial problem is NP-hard [6]. In this paper, we work in a spatially continuous setting, avoiding typical problems of graph-based discretization like anisotropy and metrication errors [17].

2.2. Continuous approaches. Continuous approaches address the multi-label problem by means of *convex relaxation*. To this end, the original non-convex energy is replaced with a convex lower bound, which can be minimized globally. We automatically get a bound on the solution and know how far we are at most from the global optimum. How good the bound is depends on the *tightness* of the relaxation, i.e. how close the energy is to the convex envelope relaxation. For particular labeling problems, this strategy even leads to globally optimal solutions. For example, as in the discrete setting, it is possible to solve the two-label problem with length regularity, i.e. the regularizer being equal to the length of the interface between the two regions, in a globally optimal way [23].

The framework presented in this paper is based on the calibration or lifting idea for the Mumford-Shah functional, which was analyzed in depth in [1, 2]. The idea is that rather than optimizing for the original labeling function, one instead estimates the characteristic function of its epigraph (called the subgraph in [2]). Thus, one ends up with a relaxation of the original problem in terms of these characteristic functions, which is convex. The question is whether the solution of the relaxation corresponds to a solution of the original problem. In [28, 27], it was shown that one can achieve a globally optimal solution for the special case of a linearly ordered set of labels and convex regularizers. This construction is related to the discrete approach proposed in [16].

For the general multi-label case, however, there is no relaxation known that would lead to provably optimal solutions. Relaxations of different tightness have been proposed in [21, 9, 41]. They all have in common that they are very memory intensive if the number of labels becomes larger. This makes it impossible to use them for scenarios with thousands of labels, like for example optic flow.

Our previous conference publication [14] relied on a straight-forward relaxation of the regularizer proposed in [41] and further generalized in [21], which was designed for a discrete label space. In this paper, we instead employ the relaxation proposed in [25], which is based on the calibration method [2]. Not only is it provably tighter than the above, but it also allows us to accurately represent regularizers for continuous label spaces. In this context, we show that the discrete relaxation [21, 14] can also be interpreted as a special case of the more general continuous framework.

3. Multi-dimensional Label Spaces.

3.1. Discrete product label spaces. From now on we assume that the space of labels is a product of a finite number d of spaces, $\Gamma = \Lambda_1 \times \cdots \times \Lambda_d$. In order to give a more visual explanation of the main idea behind our work, we first discuss the *discrete* case, where $|\Lambda_k| = N_k \in \mathbb{N}$.

The convex relaxation introduced in [21, 41] works as follows. Instead of looking for a labeling $\boldsymbol{u} : \Omega \to \Gamma$ directly, we associate each label γ with a binary indicator function $u^\gamma \in \mathcal{L}^2(\Omega, \{0,1\})$, where $u^\gamma(x) = 1$ if and only if $\boldsymbol{u}(x) = \gamma$. To make sure that a unique label is assigned to each point, only one of the indicator functions can have the value one. We can model this restriction by viewing $\boldsymbol{u}$ as a function mapping

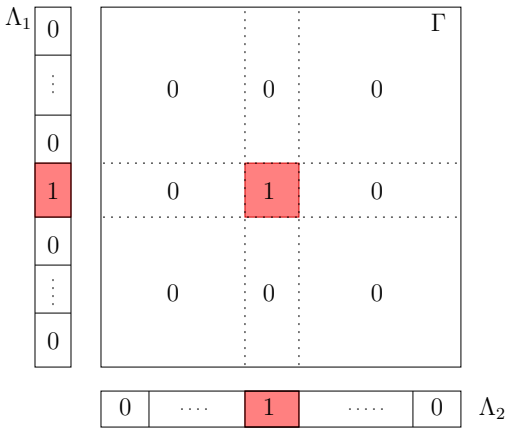


Fig. 3.1: *The central idea of the reduction technique is that if a single indicator function in the product space* Γ *takes the value* 1*, then this is equivalent to setting an indicator function in each of the factors* Λ_j*. The memory reduction stems from the fact that there are much more labels in* Γ *than in all the factors* Λ_j *combined.*

into the set of corners Δ of the N-simplex:

$$\boldsymbol{u} \in \mathcal{L}^2(\Omega, \Delta) \text{ with } \Delta = \left\{ \boldsymbol{x} \in \{0,1\}^N \ : \ \sum_{j=1}^{N} x_j = 1 \right\}. \tag{3.1}$$

Obviously, we can identify $\boldsymbol{u}$ with the vector $(u^\gamma)_{\gamma\in\Gamma}$ of indicator functions. Let $\langle\cdot,\cdot\rangle$ denote the inner product on the Hilbert space $\mathcal{L}^2(\Omega)$, then problem (1.1) can thus be written in the equivalent form

$$\operatorname*{argmin}_{\boldsymbol{u}\in\mathcal{L}^2(\Omega,\Delta)} \left\{ J(\boldsymbol{u}) + \sum_{\gamma\in\Gamma} \langle u^\gamma, c^\gamma \rangle \right\}, \tag{3.2}$$

where we use bold face notation $\boldsymbol{u}$ for vectors $(u^\gamma)_{\gamma\in\Gamma}$ indexed by elements in Γ. We write $c^\gamma(x) := c(x,\gamma)$ for the discrete data term. We use the same symbol J to also denote the regularizer on the reduced space. Its definition requires careful consideration, and will be discussed in detail later.

The central idea of the paper is the following. The full discrete label space Γ has $N = N_1 \cdot \ldots \cdot N_d$ elements, which means that it requires N indicator functions to represent a labeling, one for each label. We will show that it suffices to use $N_1 + \ldots + N_d$ indicator functions, which is a considerable reduction in problem dimensionality, thus computation time and memory requirements. We achieve this by replacing the indicator functions on the product Γ by indicator functions on the components Λ_k.

To this end, we associate to each label $\lambda \in \Lambda_k, 1 \le k \le d$ an indicator function v_k^λ. In each component k, only one of the indicator functions can be set. Thus, the vector $\boldsymbol{v}_k = (v_k^\lambda)_{\lambda\in\Lambda_k}$ which consists of N_k binary functions can be viewed as a mapping into the corners of the simplex Δ_k,

$$\boldsymbol{v}_k \in \mathcal{L}^2(\Omega, \Delta_k) \text{ with } \Delta_k = \left\{ \boldsymbol{x} \in \{0,1\}^{N_k} \ : \ \sum_{j=1}^{N_k} x_j = 1 \right\}. \tag{3.3}$$

In particular, the reduced set of indicator functions $\boldsymbol{v} = (v_k^\lambda)_{1 \leq k \leq d, \gamma \in \Lambda_k}$ can be seen as a map $\mathcal{L}^2(\Omega, \Delta_\times)$ with

$$\Delta_\times = \Delta_1 \times \ldots \times \Delta_d \subset \mathbb{R}^{N_1 + \ldots + N_d} \,. \tag{3.4}$$

Note that an element $\boldsymbol{v} \in \mathcal{L}^2(\Omega, \Delta_\times)$ consists indeed of exactly $N_1 + \ldots + N_d$ binary functions.

The following proposition illuminates the relationship between the original space of indicator functions $\mathcal{L}^2(\Omega, \Delta)$ and the reduced indicator function space $\mathcal{L}^2(\Omega, \Delta_\times)$, which is easy to understand visually, see Figure 3.1.

Proposition 3.1. *A bijection $\boldsymbol{v} \mapsto \boldsymbol{u}$ from $\mathcal{L}^2(\Omega, \Delta_\times)$ onto $\mathcal{L}^2(\Omega, \Delta)$ is defined by setting*

$$u^\gamma := v_1^{\gamma_1} \cdot \ldots \cdot v_d^{\gamma_d} \,, \quad \forall \gamma = (\gamma_1, \ldots, \gamma_d) \in \Gamma. \tag{3.5}$$

The proof of this proposition as well as of the other following results is given in the appendix.

Using this reduced function space, another equivalent formulation to (1.1) and (3.2) can be given as

$$\operatorname*{argmin}_{\boldsymbol{v} \in \mathcal{L}^2(\Omega, \Delta_\times)} \left\{ J(\boldsymbol{v}) + \sum_{\gamma \in \Gamma} \langle v_1^{\gamma_1} \cdot \ldots \cdot v_d^{\gamma_d}, c^\gamma \rangle \right\} . \tag{3.6}$$

While we have reduced the dimensionality of the problem considerably, we have introduced another difficulty: the data term is not convex anymore, since it contains a product of the components. Thus, in the relaxation, we need to take additional care to make the final problem convex again.

3.2. Continuous label spaces and relaxation framework. We now turn to the more general case that each factor Λ_k is an interval in $\mathbb{R}$, which means that we deal with a continuous label space with an infinite number of labels. In this situation, one is also interested in a number of continuous regularizers, which cannot be modeled satisfyingly on a discrete label space. As in the discrete case, the regularizers are usually not convex and require a relaxation.

In the context of continuous labeling problems where the label range is an interval, a central idea is *functional lifting*, which is a variant of the calibration method [2]. Here, one works with characteristic functions describing the hypograph instead of the labeling function itself, an idea that was further refined and applied to a variety of image processing problems in a number of subsequent works [9, 25, 26, 27]. We are going to translate this framework to the case of a product label space. With the regularizer, we can restrict ourselves to the case that it can be decomposed into the sum of regularizers on each component. However, for the data term this is not possible since the cost function usually cannot be decomposed in a similar way. Therefore, we need to define a relaxation framework in which we still can express arbitrary cost functions.

Let us first consider a single component $u_k : \Omega \to \Lambda_k$ of the full labeling function $\boldsymbol{u}$. The characteristic function of its hypograph is defined on $\Omega \times \Lambda_k$ as

$$1_{\mathrm{hyp}(u_k)}(x, \lambda) = \begin{cases} 1 & \text{if } \lambda < u_k(x), \\ 0 & \text{else.} \end{cases} \tag{3.7}$$

In [2, 26, 27], the labeling problem is reformulated in terms of new unknowns which correspond to these characteristic functions. The reason is equation (4.1), which we discuss later and which allows to give a convex reformulation of the regularizer in terms of the new unknowns. This allows us to obtain a globally optimal solution in the new variables, which often is at least close to and sometimes equal to the solution of the original non-convex problem.

In our case, however, we need different variables in order to be able to simultaneously formulate a convex relaxation of the data term. We work with the *indicator functions* denoting if a specific label λ is set at a point $x \in \Omega$, related to a labeling $\boldsymbol{u}$ by

$$v_k(x, \lambda) = \delta(u_k(x) - \lambda), \tag{3.8}$$

where δ is the Dirac distribution. Note that the new unknowns are actually distributions on the higher dimensional space $\Omega \times \Lambda_k$, which however will reduce to regular functions after discretization. They serve as a generalization of the discrete label indicator functions $\boldsymbol{v}_k \in \mathcal{L}^2(\Omega, \Delta_k)$ in (3.3) to the continuous case, in particular they satisfy the relations

$$\int_{\Lambda_k} v_k(x, \lambda) \, \mathrm{d}\lambda = 1, \quad \int_{\Lambda_k} \lambda \, v_k(x, \lambda) \, \mathrm{d}\lambda = u_k(x), \tag{3.9}$$

which mimic the discrete case with sums replaced by integrals. Intuitively, this means that for each fixed $x \in \Omega$, $v_k(x, \cdot)$ has a total mass of 1 and is concentrated on the label $u_k(x) \in \Lambda_k$.

We will reformulate the labeling problem in terms of the new variables $\boldsymbol{v}$ in Section 4. Some things have to be kept in mind, however. Since the new variables are distributions in the continuous case, we cannot formulate a well-defined minimization problem without first reducing them to $\mathcal{L}^2$-functions . This means that before writing down the actual minimization problem we want to solve in the new variables, we have to introduce a discretization of the label space. Despite the necessary discretization, we follow other previous works which employ the lifting idea [25, 26, 27, 28], and still insist that we correctly deal with a continuous label space. This is justified since the definition of the continuous regularizers in Section 5 does not make use of the discretization, in contrast to e.g. [21], where the label space is discretized from the beginning. One thing which remains to be discussed, however, is whether the discrete solutions converge to the continuous one when the label space discretization is refined, in the spirit of [8]. This is a possible avenue for future work.

3.3. Regularization. As stated in the beginning, we consider a *separable* regularizer of the form

$$J(\boldsymbol{u}) = \sum_{k=1}^{d} J_k(u_k). \tag{3.10}$$

In order to define its components, we require some technical preliminaries. Recall [3, Definition 10.5.1] that for functions u_k in the space $\mathcal{SBV}(\Omega)$ of special functions of bounded variation, the distributional derivative Du_k can be decomposed as

$$Du_k = \nabla u_k \, \mathrm{d}x + (u_k^+ - u_k^-)\nu_{u_k} \, \mathrm{d}\mathcal{H}^{n-1} \llcorner S_{u_k} \tag{3.11}$$

into a differentiable part and a jump part, see Figure 3.2. Here, S_{u_k} is the $(n-1)$-dimensional jump set of u_k, where the values jump from u_k^- to u_k^+, ν_{u_k} is the normal

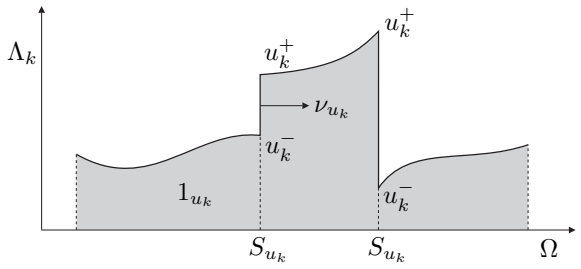


Fig. 3.2: *A special function of bounded variation u has an approximate gradient everywhere except on a nullset S_u, where the values jump from u^- to u^+. The normal ν_u denotes the direction of the jump from small to large values.*

to S_{u_k} oriented towards the u_k^+ side, and ∇u_k is the approximate gradient of u_k [3, Proposition 10.4.1]. The measure $\mathcal{H}^{n-1} \llcorner S_{u_k}$ is the $(n-1)$-dimensional Hausdorff measure restricted to the set S_{u_k}. We refer to [3] for a comprehensive introduction to functions of bounded variation.

Making use of this decomposition, we can introduce the framework for regularization. We consider regularizers for problem (1.1) of the form (3.10), with

$$J_k(u_k) = \int_{\Omega \setminus S_{u_k}} h_k(x, u_k(x), \nabla u_k(x)) \, \mathrm{d}x + \int_{S_{u_k}} d_k\big(s, u_k^-(s), u_k^+(s)\big) \, \mathrm{d}\mathcal{H}^{n-1}(s), \tag{3.12}$$

with functions $h_k : \Omega \times \Lambda_k \times \mathbb{R}^n \to \mathbb{R}$ and $d_k : \Omega \times \Lambda_k \times \Lambda_k \to \mathbb{R}$. The functions h_k and d_k have to satisfy the following conditions:

1. $h_k(x, \lambda, \cdot)$ is convex for fixed $x \in \Omega, \lambda \in \Lambda_k$, and $h_k(x, \cdot, \cdot)$ is lower semicontinuous for fixed $x \in \Omega$.
2. $d_k(x, \cdot, \cdot)$ is a metric on Λ_k for fixed $x \in \Omega$, and d_k is continuous.

The interesting task, of course, is to identify suitable choices of h_k and d_k, and to interpret what the choice means in practice. We will turn to this in Section 5. Before we can explore the possible regularizers, however, we need to introduce a convex relaxation of the general regularizer (3.12) in Section 4.

3.4. Notation conventions. Because the label space is multi-dimensional, the notation requires multiple indices and is slightly more complex. Throughout this work, we keep the following conventions to keep it as clear as possible. The index $k = 1, \ldots, d$ enumerating the factors of the product space is always written as a subscript. Indices which are Greek letters always enumerate labels, where γ, χ are labels in the full product space Γ with components $\gamma_k, \chi_k \in \Lambda_k$. Greek letters λ, μ denote labels in one of the factors Λ_k. If the label space is discrete or has been discretized, the label is written as a superscript to the indicator functions v_k^λ. In the case of a continuous label space, the indicator functions v_k live on $\Omega \times \Lambda_k$, thus the label appears as an argument of the function $v_k(x, \lambda)$.

4. Convex relaxation. The minimization problem (3.2) which we want to solve is not convex: neither is the energy a convex function nor is the domain of minimization a convex set. Thus, the task of finding a global minimizer is in general computationally infeasible. We therefore propose a *convex relaxation*. This means that instead of minimizing the original functional, we minimize a convex one (ideally, the exact convex envelope) over the convex hull of the original domain.

The relaxation is defined in terms of the new variables v_k defined in (3.8). After obtaining a solution $\hat{\boldsymbol{v}}$, the question remains of whether the solution corresponds to a function $\hat{\boldsymbol{u}}$ which solves the original problem. In general, this is not the case, but we can compute a projection $\Pi(\hat{\boldsymbol{v}})$ onto the original problem domain and obtain an optimality bound. Indeed, the energy of the optimal solution $\hat{\boldsymbol{u}}$ must lie somewhere between the energies of $\hat{\boldsymbol{v}}$ and $\Pi(\hat{\boldsymbol{v}})$, as $\hat{\boldsymbol{v}}$ minimizes the relaxation and $\Pi(\hat{\boldsymbol{v}})$ lies in the original problem domain in which $\hat{\boldsymbol{u}}$ is a minimizer.

In the following subsection we will introduce first a convex relaxation of the regularizer, which is based on the calibration method—however, our variables are different from the ones used in previous work, which requires a slight reformulation. Thereafter, we present the new convex relaxation of the data term and show how it is an improvement over the one presented in the original conference paper [14].

4.1. Convex relaxation of the regularizer. Our first goal is to give a new representation of the regularizer defined in (3.12). While in general it is not convex in the labeling $\boldsymbol{u}$, we will obtain a representation which is convex in the new variables v_k defined in (3.8). We do this by making use of the calibration or lifting technique described in detail in [2]. Lemma 3.9 in [2] states that under the previous assumptions on h_k and d_k, the regularizer J_k for each component can be represented as

$$J_k(u_k) = \sup_{\boldsymbol{\phi} \in K} \left\{ \int_{\Omega \times \Lambda_k} \boldsymbol{\phi}^1 \cdot \nabla_x 1_{\mathrm{hyp}(u_k)} + \phi^2 \, \partial_\lambda 1_{\mathrm{hyp}(u_k)} \, \mathrm{d}(x, \lambda) \right\} \tag{4.1}$$

with the convex set

$$\begin{aligned} K = \Bigg\{ \boldsymbol{\phi} = (\boldsymbol{\phi}^1, \phi^2) \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R}) \text{ such that for all } x \in \Omega \text{ and } \lambda, \mu \in \Lambda_k, \\ \phi^2(x, \lambda) \geq h_k^*(x, \lambda, \boldsymbol{\phi}^1(x, \lambda)) \text{ and } \left| \int_\lambda^\mu \boldsymbol{\phi}^1(x, s) \, \mathrm{d}s \right| \leq d_k(x, \lambda, \mu) \Bigg\}. \end{aligned} \tag{4.2}$$

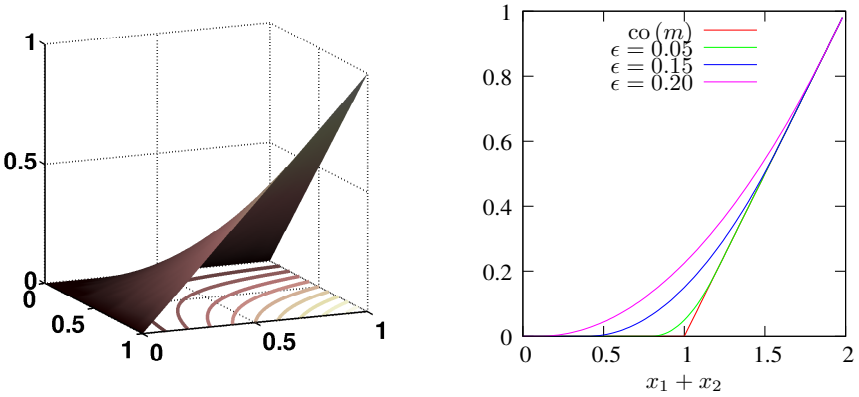
Above, $h_k^*(x, \lambda, \cdot)$ denotes the convex conjugate of $h_k(x, \lambda, \cdot)$. In a slight abuse of notation, the index c in the set $C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R})$ indicates that ϕ must have compact support, but only w.r.t. the $x \in \Omega$ variable. Note that (4.1) is a *convex* representation of the regularizer in terms of the characteristic functions $1_{\mathrm{hyp}(u_k)}$ of the hypograph of u_k, see equation (3.7). However, what we want is a convex representation in terms of our new unknowns v_k. We give this reformulation in the following theorem.

Theorem 4.1. *Let J_k be of the form (3.12), and the indicator functions v_k defined as in (3.8). Then*

$$J_k(u_k) = \sup_{(\boldsymbol{p}, b) \in C_k} \left\{ \int_{\Omega \times \Lambda_k} (-\mathrm{div}(\boldsymbol{p}) - b) \, v_k \, \mathrm{d}(x, \lambda) \right\}, \tag{4.3}$$

with the convex set

$$\begin{aligned} C_k = \Big\{ (\boldsymbol{p}, b) \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R}) \text{ such that for all } x \in \Omega \text{ and } \lambda, \mu \in \Lambda_k, \\ b(x, \lambda) \geq h_k^*\big(x, \lambda, \partial_\lambda \boldsymbol{p}(x, \lambda)\big), \\ |\boldsymbol{p}(x, \lambda) - \boldsymbol{p}(x, \mu)|_2 \leq d_k(x, \lambda, \mu) \Big\}. \end{aligned} \tag{4.4}$$



(a) Product function $m(x_1, x_2) = x_1 x_2$

(b) Convex envelope co (m) and mollified versions for different ϵ

Fig. 4.1: *Product function and its mollified convex envelope for the case* $d = 2$.

Note that similarly to the discrete version of the indicator functions, the discrete version of the set C_k in (4.4) will consist of tuples $(\boldsymbol{p}^\lambda, b^\lambda)_{\lambda \in \Lambda_k}$ of functions. Taking a closer look at equation (4.3), we can see that the right hand side is a convex functional in the new variables v_k. Thus, we have achieved our goal and can turn towards finding a similar relaxation of the data term.

4.2. Convex relaxation of the data term. In this subsection, we deal with the non-convexity of the data term in (3.6),

$$E_{\text{data}}(\boldsymbol{v}) = \sum_{\gamma \in \Gamma} \langle v_1^{\gamma_1} \cdot \ldots \cdot v_d^{\gamma_d}, c^\gamma \rangle \ . \tag{4.5}$$

Specifically, we show two different ways how it can be replaced with a convex function which coincides with the original data term for binary functions. We first describe the convexification idea from the original conference paper [14] in the discrete case with a label space of dimension $d = 2$. While it leads to a working relaxation, it has certain shortcomings, the main problem being that an unwanted constant solution has to be avoided by additional smoothing when moving on from binary to continuous functions. These shortcomings will be remedied by a new relaxation technique which we explain thereafter. We will show that this relaxation is actually the best possible one, i.e. the convex envelope of the data term. Note that for the data term, we already work in the setting of a discretized label space. While it is possible to give a well-defined theoretical justification of the relaxation for the continuous case, the associated trouble and loss of clarity is not worth the small theoretical gain.

Discrete two-dimensional case. In [14], we suggested to replace the multiplication function $m(v_1^{\gamma_1}, \ldots, v_d^{\gamma_d}) := v_1^{\gamma_1} \cdot \ldots \cdot v_d^{\gamma_d}$ with its convex envelope co (m). Analyzing the epigraph of m, see Figure 4.1(a), shows that

$$\text{co}\,(m)\,(v_1^{\gamma_1}, \ldots, v_d^{\gamma_d}) = \begin{cases} 1 & \text{if } v_1^{\gamma_1} = \ldots = v_d^{\gamma_d} = 1, \\ 0 & \text{if any } v_k^{\gamma_k} = 0. \end{cases} \tag{4.6}$$

This means that if in the functional, m is replaced by the convex function co (m), we retain the same binary solutions, as the function values on binary input are the same.

We lose nothing on first glance, but on second glance, we forfeited differentiability of the data term, since $\operatorname{co}(m)$ is not a smooth function anymore. Furthermore, the new function we obtain is not the correct convex envelope of the full data term, but only for the constituting addends. The particular problem this leads to is that for the constant function $\hat{\boldsymbol{v}}$ defined by

$$\hat{v}_k^\lambda(x) := 1/N_k \tag{4.7}$$

the energy of the data term and hence the total energy is zero.

In [14], this problem was circumvented by an additional mollification of the convex envelope. We replaced $\operatorname{co}(m)$ again by a mollified function $\operatorname{co}(m)_\epsilon$, where $\epsilon > 0$ is a small constant. We illustrate this for the case $d = 2$, where one can easily write down the functions explicitly. In this case, the convex envelope of multiplication is

$$\operatorname{co}(m)(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 + x_2 \leq 1, \\ x_1 + x_2 - 1 & \text{otherwise.} \end{cases} \tag{4.8}$$

This is a piecewise linear function of the sum of the arguments, i.e symmetric in x_1 and x_2, see Figure 4.1(b). We smoothen the kink by replacing $\operatorname{co}(m)$ with smoothed version $\operatorname{co}(m)_\epsilon$, see [14]. This function does not satisfy the envelope condition (4.6) exactly, but only fulfills the less tight

$$\operatorname{co}(m)_\epsilon(x_1, \ldots, x_d) \begin{cases} = 1 & \text{if } x_1 = \cdots = x_d = 1, \\ \leq \epsilon & \text{if any } x_j = 0. \end{cases} \tag{4.9}$$

Notably, the data term energy of the constant trivial minimizer (4.7) becomes now $\epsilon \sum_\gamma c^\gamma$, which means that the relaxation of the data term leads to the correct pointwise solution with energy $\min_\gamma(c^\gamma)$ if $\epsilon > \min_\gamma(c^\gamma)/\sum_\gamma c^\gamma$. Since the condition must be satisfied for each point $x \in \Omega$, it is best to let $\epsilon = \epsilon(x)$ depend on $x \in \Omega$ and set it pointwise to the minimal possible value. However, the choice of mollified envelope is suboptimal since it is just an approximation to the correct envelope and distorts the original problem. Thus, we are now going to propose a novel relaxation of the data term which avoids this problem altogether and is easier to deal with in higher dimensional label spaces.

New convex envelope relaxation for the general d-dimensional case. In this paragraph, we describe our new relaxation of the data term. It is the tightest possible relaxation and does not suffer from the described drawbacks of the relaxation in [14]. The new relaxation of $E_{\text{data}}(\boldsymbol{v})$ is one of the main additional contributions of this paper. It is defined as

$$R_{\text{data}}(\boldsymbol{v}) := \sup_{\boldsymbol{q} \in \mathcal{Q}} \left\{ \int_\Omega \sum_{\gamma_1 \in \Lambda_1} q_1^{\gamma_1} v_1^{\gamma_1} + \ldots + \sum_{\gamma_d \in \Lambda_d} q_d^{\gamma_d} v_d^{\gamma_d} \, \mathrm{d}x \right\}. \tag{4.10}$$

The additional dual variables $\boldsymbol{q} = (\boldsymbol{q}_k)_{k=1..d}$ range over the convex set

$$\begin{aligned} \mathcal{Q} := \Big\{ \boldsymbol{q} \in \mathcal{L}^2(\Omega, \mathbb{R}^{N_1 + \ldots + N_d}) \text{ such that for all } x \in \Omega \text{ and } \gamma \in \Gamma, \\ q_1^{\gamma_1}(x) + \ldots + q_d^{\gamma_d}(x) \leq c^\gamma(x) \Big\}. \end{aligned} \tag{4.11}$$

We first establish that the relaxation coincides with the original energy for binary functions.

Proposition 4.2. *Let $\boldsymbol{v} \in \mathcal{L}^2(\Omega, \Delta_\times)$ be a binary function representing the label $\gamma(x) \in \Gamma$ at each point $x \in \Omega$. Then*

$$R_{data}(\boldsymbol{v}) = \int_\Omega c^{\gamma(x)}(x)\,\mathrm{d}x = E_{data}(\boldsymbol{v}). \tag{4.12}$$

In addition, one can prove the following theorem, which shows that the relaxation of the data term has the correct pointwise minimizers, in contrast to the one proposed in [14]. This means that no smoothing is necessary and an exact minimization algorithm can be employed to obtain solutions.

Theorem 4.3. *Let $\hat{\boldsymbol{v}} \in \mathcal{L}^2(\Omega, \Delta_\times)$ be a binary minimizer of E_{data}. Then $\hat{\boldsymbol{v}}$ is also a minimizer of the relaxation,*

$$\hat{\boldsymbol{v}} \in \operatorname*{argmin}_{\boldsymbol{v} \in \mathcal{L}^2(\Omega, \mathrm{co}(\Delta_\times))} \{R_{data}(\boldsymbol{v})\}. \tag{4.13}$$

In particular, $E_{data}(\hat{\boldsymbol{v}}) = R_{data}(\hat{\boldsymbol{v}}) = \int_\Omega \hat{c}(x)\,\mathrm{d}x$ with $\hat{c}(x) := \inf_{\gamma\in\Gamma}(c^\gamma(x))$ for $x \in \Omega$.

In fact, it turns out that the proposed data term relaxation is the best possible one, being the convex envelope of the data term as stated in the proposition below. More specifically, this is up to the natural sum equality $\sum_{\lambda\in\Lambda_k} v_k^\lambda = 1$, which is the first equation in (3.9) and is also used in the definition (6.1) of the $\boldsymbol{v}$ domain in the overall optimization problem. To make this statement precise, we first need a general definition of the data term E_{data} for *all* indicator functions $\boldsymbol{v} \in \mathcal{L}^2(\Omega, \mathbb{R}^{N_1+\ldots+N_d})$, and not only for binary ones. If $\boldsymbol{v}$ is binary representing the label $\gamma(x) \in \Gamma$ at each $x \in \Omega$, i.e. $v_k^\lambda(x) = \chi_{\lambda=\gamma_k(x)}$, E_{data} is already defined by (4.5) as

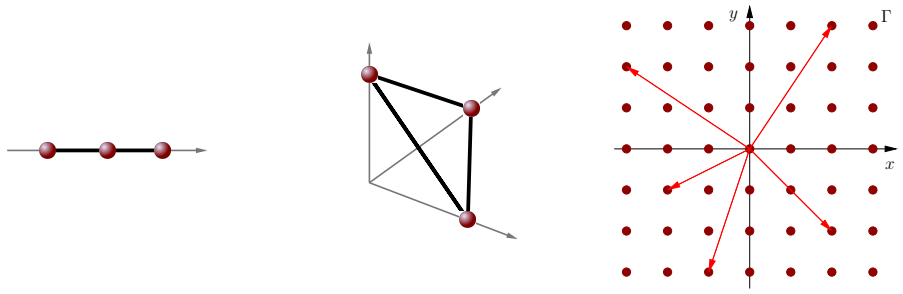
$$E_{\mathrm{data}}(\boldsymbol{v}) = \int_\Omega c^{\gamma(x)}(x)\,\mathrm{d}x = \int_\Omega \sum_{\gamma\in\Gamma} c^\gamma(x)\, v_1^{\gamma_1}(x)\cdots v_d^{\gamma_d}(x)\,\mathrm{d}x. \tag{4.14}$$

For all other $\boldsymbol{v}$ we set $E_{\mathrm{data}}(\boldsymbol{v}) := \infty$.

Proposition 4.4. *The convex envelope of $E_{data}(\boldsymbol{v})$ is given by $R_{data}(\boldsymbol{v}) + \delta_{\mathcal{S}}(\boldsymbol{v})$ with $\mathcal{S} := \left\{\boldsymbol{v} \,\middle|\, \sum_{\lambda\in\Lambda_k} v_k^\lambda(x) = 1 \quad \forall\ 1 \le k \le d,\ x \in \Omega\right\}$.*

5. Multilabel Regularizers. In this section, we will explore suitable choices of the regularizer, and how they fit within the proposed framework. In particular, we will see how our model can be specialized to the case of discrete label spaces where the label distance has an Euclidean representation. This special case was discussed in [21, 14], and we will see that our framework leads to a tighter relaxation for this case. We will also discuss additional continuous regularizers which become possible based on the lifting framework discussed in the last section. These were introduced in the previous works [9, 26, 27] when the unknowns were the characteristic functions of the hypographs of u_k. We show how we can accommodate them to depend on the new unknowns. Notably, in each dimension of the label space its own type of regularization can be chosen, in particular discrete and continuous regularizers can be mixed freely.

5.1. Discrete label space and its Euclidean representation. We first consider the special case of a discrete label space Λ_k. Thus, we need to define a regularizer $J_k : \mathcal{L}^2(\Omega, \mathrm{co}\,(\Delta_k)) \to \mathbb{R}$ for functions with values in the convex hull of the



(a) Ordered embedding (b) Potts embedding (c) Optic flow embedding

Fig. 5.1: *Different embeddings for a label space. In an ordered embedding, all labels are mapped onto a line, while for the Potts model, every label is mapped onto a different unit vector. For optical flow, each label is already a vector in* $\mathbb{R}^2$*, so a sensible embedding is given by the identity.*

simplex Δ_k. We first present the construction used in [21, 14], and then show how we can embed it into our more general framework.

We assume that the metric d_k has an *Euclidean* representation. This means that each label $\lambda \in \Delta_k$ shall be *represented* by an M_k-dimensional vector $\boldsymbol{a}_k^\lambda \in \mathbb{R}^{M_k}$ with a $M_k \geq 1$, and the distance d_k is defined as the Euclidean distance between the representations,

$$d_k(\lambda, \mu) = \left|\boldsymbol{a}_k^\lambda - \boldsymbol{a}_k^\mu\right|_2 \text{ for all } \lambda, \mu \in \Delta_k \,. \tag{5.1}$$

The goal in the construction of J_k is that the higher the distance between labels and the longer the jump set, the higher shall be the penalty imposed by J_k. To make this idea precise, we introduce the linear mappings $A_k : \operatorname{co}(\Delta_k) \to \mathbb{R}^{M_k}$ which map labels onto their representations,

$$A_k(\lambda) = \boldsymbol{a}_k^\lambda \text{ for all } \lambda \in \Delta_k \,. \tag{5.2}$$

When the labels are enumerated and represented by the indicator functions $\boldsymbol{v}_k$ in (3.3), then in matrix notation, the vectors $\boldsymbol{a}_k^\lambda$ become exactly the columns of A_k, which shows the existence of this map. It turns out that a regularizer with desirable properties can be defined by

$$J_k^A(\boldsymbol{v}_k) := \mathrm{TV}_\mathrm{v}(A_k \boldsymbol{v}_k) \,, \tag{5.3}$$

where

$$\mathrm{TV}_\mathrm{v}(\boldsymbol{f}) := \int_\Omega \sqrt{\sum_{i=1}^m |\nabla f_i|_2^2}\, \mathrm{d}x \tag{5.4}$$

denotes the vectorial total variation for functions $\boldsymbol{f} : \Omega \to \mathbb{R}^m$ taking values in a real vector space of dimension m. The following theorem has been proven in [21] and shows why the above definition makes sense.

Theorem 5.1. *The regularizer* J_k^A *defined in* (5.3) *has the following properties:*

1. J_k^A *is convex and positively homogeneous on* $\mathcal{L}^2(\Omega, \operatorname{co}(\Delta_k))$.

2. $J_k^A(\boldsymbol{v}_k) = 0$ *for any constant labeling* $\boldsymbol{v}_k$.
3. *If* $S \subset \Omega$ *has finite perimeter* $\mathrm{Per}(S)$*, then for all labels* $\lambda, \mu \in \Lambda_k$*,*

$$J_k^A(\lambda 1_S + \mu 1_{S^c}) = d_k(\lambda, \mu)\,\mathrm{Per}(S)\,, \tag{5.5}$$

i.e. a change in labels is penalized proportionally to the distance between the labels and the perimeter of the interface.

For the sake of simplicity, we only give the main examples for distances with Euclidean representations. More general classes of distances on the labels can also be used, see [21].

- The case of ordered labels, where the embedding follows the natural ordering of $\lambda, \mu \in \mathbb{R}$, Figure 5.1(a), for example by setting simply $\boldsymbol{a}_k^\lambda = \lambda$. If $d = 1$, then this case can be solved in a globally optimal way using the lifting method [27].
- The Potts or uniform distance, where $d_k(\lambda, \mu) = 1$ if and only if $\lambda = \mu$, and zero otherwise. This distance function can be achieved by setting $\boldsymbol{a}_k^\lambda = \frac{1}{\sqrt{2}}\boldsymbol{e}^\lambda$, where $(\boldsymbol{e}^\lambda)_{\lambda \in \Lambda_k}$ is an orthonormal basis in $\mathbb{R}^{N_k}$, see Figure 5.1(b). All changes between labels are penalized equally.
- Another typical case is that the $\boldsymbol{a}_k^\lambda$ denote feature vectors or actual geometric points, for which $|\cdot|_2$ is a natural distance. For example, in the case of optic flow, each label corresponds to a flow vector in $\mathbb{R}^2$, see Figure 5.1(c). The representations $\boldsymbol{a}_1^\lambda, \boldsymbol{a}_2^\mu$ are just real numbers, denoting the possible components of the flow vectors in x and y-direction, respectively. The Euclidean distance is a sensible distance on the components to regularize the flow field, corresponding to the regularizer of the TV-L^1 functional in [43]. Optic flow (and other geometric kinds of labels) would however more naturally be modeled with a continuous label space using one of the continuous regularizers in the later subsections.

5.2. New relaxation for the discrete label space. We will now show how to formulate the regularizer J_k^A defined above in the new more general framework. While the previous formulation (5.3) already yields a relaxation to non-binary functions $\boldsymbol{v}$, we will see that our framework results in a provably tighter one.

Taking a look at Theorem 5.1, we see that the regularizer must penalize the length of the jump set weighted by the label distance. Thus, our general regularizer in (3.12) must reduce to

$$J_k(u_k) = \int_{S_{u_k}} d_k\big(u_k^-, u_k^+\big)\,\mathrm{d}\mathcal{H}^{n-1}. \tag{5.6}$$

where d_k is the same metric as used above in the representation (5.1). We can see that in order to reduce the general form to the one above, we must enforce a piecewise constant labeling, since the approximate gradient ∇u_k must be constantly zero outside the jump set. Applying Theorem 4.1 we can find a convex representation of J_k in terms of the variables $\boldsymbol{v}$, which we formulate in the following proposition in its discretized form.

Proposition 5.2. *A convex representation of* (5.6) *in terms of the variables* $\boldsymbol{v}$ *is given by*

$$J_k(u_k) = \sup_{\boldsymbol{p} \in C_k} \left\{ \sum_{\lambda \in \Lambda_k} \int_\Omega v_k^\lambda\, \mathrm{div}\,\big(\boldsymbol{p}^\lambda\big)\,\mathrm{d}x \right\}, \tag{5.7}$$

with

$$C_k = \Big\{ \boldsymbol{p} : \Omega \times \Lambda_k \to \mathbb{R}^n \ : \boldsymbol{p}^\lambda \in C_c^1(\Omega; \mathbb{R}^n) \text{ for all } \lambda \in \Lambda_k \text{ and } \\ \left|\boldsymbol{p}^\lambda - \boldsymbol{p}^\mu\right|_2 \leq d_k(\lambda, \mu) \text{ for all } \lambda, \mu \in \Lambda_k \Big\}. \tag{5.8}$$

We can now establish the relationship between our framework and the regularizer J_k^A derived from a representation of the labels in (5.3), and show that ours is tighter.

Proposition 5.3. *Let the regularizer J_k be defined by the relaxation on the right hand side in equation (5.7). Then for all $\boldsymbol{v}_k \in \mathcal{L}^2(\Omega, \operatorname{co}(\Delta_k))$,*

$$J_k(\boldsymbol{v}_k) \geq J_k^A(\boldsymbol{v}_k). \tag{5.9}$$

Equality holds if $\boldsymbol{v}_k$ is binary.

The right hand side of inequality (5.9) is exactly the previous regularizer used in [14, 21]. This implies that for binary functions, the regularizers coincide, which can already be seen from representation (5.6), see Theorem 5.1. However, if we perform the relaxation to functions taking values between 0 and 1, inequality (5.9) implies that the new relaxation is more tight, leading to solutions closer to the global optimum.

We will show in the remainder of the section that in addition to handling the discrete case better, our method also can handle continuous regularizers which penalize a *smooth variation* of the labels. This is not possible with the piecewise constant approach of [21, 14] which uses vectorial total variation. For instance, our formulation is capable of representing more sophisticated regularizers such as Huber-TV and the piecewise smooth Mumford-Shah functional, as we will show in the following subsection. For the regularizers presented in the remainder of this section, relaxations have previously been proposed for the case of a one-dimensional label space in [9, 26, 27, 35]. However, the framework presented here is more general and allows to combine them freely in the different label dimensions.

5.3. Linear (TV) and truncated linear. For many applications, it is useful to penalize the difference between two label values λ and μ only up to a certain threshold, reasoning that once they are that different, it does not matter anymore how different exactly they are. This means that if $|\lambda - \mu|$ becomes greater than a certain value t, jumps from λ to μ are still penalized, but only by the constant t. Using linear penalization for small values this leads to the robust *truncated linear* regularizer [9]

$$J_k(u_k) = \int_{\Omega \setminus S_{u_k}} |\nabla u_k|_2 \, \mathrm{d}x \ + \ \int_{S_{u_k}} \min\left(t, \left|u_k^+ - u_k^-\right|\right) \mathrm{d}\mathcal{H}^{n-1}(s). \tag{5.10}$$

The constraint set (4.4) for this case is

$$C_k = \Bigg\{ (\boldsymbol{p}, b) \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R}) \text{ such that for all } \lambda, \mu \in \Lambda_k, \\ \left|\partial_\lambda \boldsymbol{p}^\lambda\right|_2 \leq 1, \quad \left|\boldsymbol{p}^\lambda - \boldsymbol{p}^\mu\right|_2 \leq t, \quad b = 0 \Bigg\}. \tag{5.11}$$

The second constraint needs to be imposed only if $|\lambda - \mu| \geq t$, since otherwise it is already implied by the first constraint. In particular, the standard linear (TV) penalizer can be implemented by letting $t \to \infty$ and only using the first constraint.

5.4. Cyclic penalizer, TV-S^1. Some applications have a cyclic or circular set of labels, for example regularization in the hue component in HSV or HSL color space. In this case, the distance between the last and first label is the same as between any other subsequent pair of labels. This form of regularization was discussed in the functional lifting setting in the recent work [35], and can be expressed in our framework by setting

$$J_k(u_k) = \int_{\Omega \setminus S_{u_k}} |\nabla u_k|_2 \, \mathrm{d}x \; + \; \int_{S_{u_k}} \min\left(u_k^+ - u_k^-, 1 - (u_k^+ - u_k^-)\right) \mathrm{d}\mathcal{H}^{n-1}(s) \quad (5.12)$$

for functions u_k with range $\Lambda_k := [0, 1)$. The corresponding constraint set in its discretized form is given by

$$C_k = \left\{ \begin{array}{l} (\boldsymbol{p}, b) \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R}) \text{ such that for all } \lambda \in \Lambda_k, \\ \qquad \left|\boldsymbol{p}^\lambda - \boldsymbol{p}^{\lambda+1}\right|_2 \leq 1, \quad b \equiv 0 \end{array} \right\}, \quad (5.13)$$

where the circularly ordered label space Λ_k is represented by integers $\{0, \dots, N_k - 1\}$ with addition modulo N_k.

5.5. Huber-TV. The TV regularization is known to produce staircasing effects in the reconstruction, i.e. the solution will be piecewise constant. While this is natural in case of a discrete label space, for continuous label spaces it impedes smooth variations of the solution. A remedy for this is replacing the norm $|\nabla u_k|_2$ of the gradient by $h_\alpha(\nabla u_k)$ with the Huber function

$$h_\alpha(z) := \begin{cases} \frac{1}{2\alpha} |z|_2^2 & \text{if } |z|_2 < \alpha, \\ |z|_2 - \frac{\alpha}{2} & \text{else} \end{cases} \quad (5.14)$$

for some $\alpha > 0$, which smooths out the kink at the origin. The Huber-TV regularizer is then defined by

$$J_k(u_k) = \int_\Omega h_\alpha(\nabla u_k) \, \mathrm{d}x \; + \; \int_{S_{u_k}} \left|u_k^+ - u_k^-\right| \mathrm{d}\mathcal{H}^{n-1}(s) \quad (5.15)$$

The limiting case $\alpha = 0$ leads to the usual TV regularization. Theorem (4.1) gives a convex representation for J_k, see also [27]. The constraint set in (4.4) is found to be

$$C_k = \left\{ \begin{array}{l} (\boldsymbol{p}, b) \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R}) \text{ such that for all } \lambda \in \Lambda_k, \\ \qquad b^\lambda \geq \frac{\alpha}{2} \left|\partial_\lambda \boldsymbol{p}^\lambda\right|_2^2, \quad \left|\partial_\lambda \boldsymbol{p}^\lambda\right|_2 \leq 1 \end{array} \right\}. \quad (5.16)$$

5.6. Piecewise smooth Mumford-Shah model. The celebrated Mumford-Shah regularizer [2, 26]

$$J_k(u_k) = \int_{\Omega \setminus S_{u_k}} \frac{1}{2\alpha} |\nabla u_k|_2^2 \, \mathrm{d}x \; + \; \nu \, \mathcal{H}^{n-1}(S_{u_k}) \quad (5.17)$$

with parameters $\alpha, \nu > 0$ allows to estimate a denoised image u_k which is *piecewise smooth*. Parameter ν can be used to easily control the total length of the jump set S_{u_k}. Bigger values of ν lead to a smaller jump set, i.e. the solution will be smooth on wider subregions of Ω. The constraint set in the convex representation of Theorem 4.1 becomes

$$C_k = \Big\{ (\boldsymbol{p}, b) \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R}) \text{ such that for all } \lambda, \mu \in \Lambda_k, \\ b^\lambda \geq \frac{\alpha}{2} \big|\partial_\lambda \boldsymbol{p}^\lambda\big|_2^2, \quad \big|\boldsymbol{p}^\lambda - \boldsymbol{p}^\mu\big|_2 \leq \nu \Big\}. \tag{5.18}$$

The limiting case $\alpha = 0$ gives the piecewise constant Mumford-Shah regularizer (also called Potts regularizer), which can also be obtained from Proposition 5.2 by setting $d_k(\lambda, \mu) = \nu$ for all $\lambda \neq \mu$. Compared to (5.3), this alternative yields a tighter, but more memory intensive relaxation for the Potts regularizer [9, 25].

6. Implementation.

6.1. Final relaxation to a convex problem. In order to transform the multilabel problem into the final form which we are going to solve, we formulate it in terms of the indicator functions v_k^λ on the discretized label space using the representation (4.3) for the regularizer and the relaxation (4.10) of the data term. Discretization of the label space is necessary now to arrive at a well-posed problem. Let us briefly summarize and review the objects we are dealing with in the final problem. The minimizer we are looking for is a vector $\boldsymbol{v} = (\boldsymbol{v}_k)_{k=1..d}$ of functions $\boldsymbol{v}_k \in \mathcal{L}^2(\Omega, \operatorname{co}(\Delta_k))$, which means that we are looking for a minimizer in a convex set $\mathcal{D}$,

$$\boldsymbol{v} \in \mathcal{D} := \big\{ \boldsymbol{v} \in \mathcal{L}^2(\Omega, \mathbb{R}^{N_1 + \ldots + N_d}) \text{ such that for all } x \in \Omega,\ \boldsymbol{v}(x) \in \operatorname{co}(\Delta_\times), \\ \text{with } \Delta_\times = \Delta_1 \times \ldots \times \Delta_d \big\}. \tag{6.1}$$

In the convex hull $\operatorname{co}(\Delta_\times) = \operatorname{co}(\Delta_1) \times \ldots \times \operatorname{co}(\Delta_d)$ each $\operatorname{co}(\Delta_k)$ is given by the same expression as in (3.3) but with the set $\{0, 1\}$ replaced by $[0, 1]$.

Let us now turn to the regularizer, which is defined via the relaxation in Theorem 4.1. The key ingredients are the convex sets C_k which depend on the kind of regularization we want to use—possible options were detailed in the last section. Let $\mathcal{C} := C_1 \times \ldots \times C_d$ denote the convex set of all regularizer dual variables, and define the linear operator $K : \mathcal{C} \to \mathcal{L}^2(\Omega, \mathbb{R}^{N_1 + \ldots + N_d})$ via

$$K(\boldsymbol{p}, \boldsymbol{b}) := \big(-\operatorname{div}(\boldsymbol{p}_k^\lambda) - b_k^\lambda\big)_{k=1..d, \lambda \in \Lambda_k}. \tag{6.2}$$

Then Theorem 4.1 in fact shows that the regularizer can be written in terms of $\boldsymbol{v}$ in the simple form

$$J(\boldsymbol{v}) = \sup_{(\boldsymbol{p}, \boldsymbol{b}) \in \mathcal{C}} \big\{ \langle K(\boldsymbol{p}, \boldsymbol{b}), \boldsymbol{v} \rangle \big\}, \tag{6.3}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product on $\mathcal{L}^2(\Omega, \mathbb{R}^{N_1 + \ldots + N_d})$. The fully relaxed problem we are going to solve can now be written as

$$\operatorname*{argmin}_{\boldsymbol{v} \in \mathcal{D}} \big\{ J(\boldsymbol{v}) + R_{\text{data}}(\boldsymbol{v}) \big\}, \tag{6.4}$$

using the relaxation R_{data} of the data term defined in (4.10). It is straightforward to prove the existence of solutions.

Proposition 6.1. *Problem* (6.4) *always has a minimizer* $\hat{\boldsymbol{v}} \in \mathcal{D}$.

Note that because of the relaxation, the solution might not be binary. If it already has values in Δ_k, we have found the global optimum of the original problem (1.1), otherwise we have to project the result back to the smaller set of binary valued functions. For this, let $\hat{\boldsymbol{v}}$ be a minimizer of the final relaxation (6.4). Thus, the functions $\hat{v}_k^\lambda$ might have values inbetween 0 and 1. In order to obtain a feasible solution to the original problem (1.1), we just project back to the space of allowed functions. The function $\hat{\boldsymbol{u}} \in \mathcal{L}^2(\Omega, \Gamma)$ closest to $\hat{\boldsymbol{v}}$ is given by setting

$$\hat{\boldsymbol{u}}(x) = \operatorname*{argmax}_{\gamma \in \Gamma} \left\{ \hat{v}_1^{\gamma_1}(x) \cdot \ldots \cdot \hat{v}_d^{\gamma_d}(x) \right\} , \tag{6.5}$$

i.e. we choose the label where the combined indicator functions have the highest value. This is the same as choosing the label by maximizing each component v_k separately:

$$\hat{\boldsymbol{u}}(x) = \gamma \quad \text{with } \gamma_k = \operatorname*{argmax}_{\lambda \in \Lambda_k} \left\{ \hat{v}_k^\lambda(x) \right\} \quad \text{for all } 1 \leq k \leq d. \tag{6.6}$$

We cannot guarantee that the solution $\hat{\boldsymbol{u}}$ is indeed a global optimum of the original problem (1.1), since there is nothing equivalent to the thresholding theorem [23] known for this kind of relaxation. However, we still can give a bound how close we are to the global optimum. Indeed, the energy of the optimal solution of (1.1) must lie somewhere between the energies of $\hat{\boldsymbol{v}}$ and $\hat{\boldsymbol{u}}$, as previously explained in the beginning of Section 4.

6.2. Numerical method. Using the representation (6.3) for J, and the definition (4.10) for the relaxation R_{data}, we can transform the final formulation (6.4) of the multilabel problem into the saddle point problem

$$\min_{\boldsymbol{v} \in \mathcal{D}} \max_{\substack{(\boldsymbol{p},\boldsymbol{b}) \in \mathcal{C} \\ \boldsymbol{q} \in \mathcal{Q}}} \left\{ \langle K(\boldsymbol{p}, \boldsymbol{b}) + \boldsymbol{q}, \boldsymbol{v} \rangle \right\} . \tag{6.7}$$

We minimize the energy (6.7) with a recent general fast primal-dual algorithm in [10], which is designed for this type of problems. The algorithm is essentially a gradient descent in the primal variable $\boldsymbol{v}$ and gradient ascent in the dual variables $\boldsymbol{p}$, $\boldsymbol{b}$ and $\boldsymbol{q}$, with a subsequent application of proximation operators, which act as generalized reprojections. In our case, these are just the usual orthogonal projections onto the respective constraint sets $\mathcal{D}$, $\mathcal{C}$ and $\mathcal{Q}$. The algorithm update equations for our case can be derived in a straightforward manner from the general "Algorithm 1" presented in [10]. The remaining question is therefore only how to compute the projections onto the constraint sets after each algorithm iteration. Since these sets are defined by numerous non-local constraints, a direct projection is quite costly.

Therefore, we suggest to implement as many constraints as possible using Lagrange multipliers by adding specific additional terms to the energy. This comes at the cost of having more terms in the final overall energy and that the optimization is done also over additional variables, the Lagrange multipliers. However, in the end less of the explicit constraints remain, so that the projections become easier to calculate. Also, the algorithm complexity remains the same since the update equations are still straightforward.

First, the simplex constraint $\boldsymbol{v} \in \mathcal{D}$, i.e. $v_k \in \operatorname{co}(\Delta_k)$ with Δ_k in (3.3) for $1 \leq k \leq d$, is enforced by adding the Lagrange multiplier terms

$$\sup_{\boldsymbol{\sigma}} \sum_{k=1}^{d} \int_{\Omega} \sigma_k(x) \left(\sum_{\lambda \in \Lambda_k} v_k^{\lambda}(x) - 1 \right) \mathrm{d}x \tag{6.8}$$

to the energy (6.7), optimizing over $\boldsymbol{\sigma} : \Omega \to \mathbb{R}^d$ in addition to the other variables. This leaves just the simple condition $\boldsymbol{v} \geq 0$ for the indicator variables $\boldsymbol{v}$. We note that it is also possible to implement the simplex constraint explicitly by the iterative algorithm [22]. However, in the end this increases the computation time per iteration many times over since the projection then requires $\mathcal{O}(N_1 + \ldots + N_d)$ steps in the worst case. Also, the explicit projection only slightly reduces the number of iterations needed to compute a minimizer of (6.7) to a certain precision. Therefore, overall the Lagrange multiplier approach turns out to be faster and is also easier to implement.

Next, we enforce the constraints $(\boldsymbol{p}, \boldsymbol{b}) \in \mathcal{C}$ on the dual variables of the regularizer by introducing new variables

$$\boldsymbol{d}_k^{\lambda} = \partial_{\lambda} \boldsymbol{p}_k^{\lambda} \qquad \text{or} \qquad \boldsymbol{d}_k^{\lambda\mu} = \boldsymbol{p}_k^{\lambda} - \boldsymbol{p}_k^{\mu}, \tag{6.9}$$

depending on the kind of constraints in C_k. To enforce these equalities, we add the corresponding Lagrange multiplier terms

$$\inf_{\boldsymbol{\eta}} \int_{\Omega} \boldsymbol{\eta}_k^{\lambda} \big(\partial_{\gamma} \boldsymbol{p}_k^{\lambda} - \boldsymbol{d}_k^{\lambda} \big) \, \mathrm{d}x \qquad \text{or} \qquad \inf_{\boldsymbol{\eta}} \int_{\Omega} \boldsymbol{\eta}_k^{\lambda,\mu} \big(\boldsymbol{p}_k^{\lambda} - \boldsymbol{p}_k^{\mu} - \boldsymbol{d}_k^{\lambda\mu} \big) \, \mathrm{d}x \tag{6.10}$$

for each $1 \leq k \leq d$ and $\lambda \in \Gamma_k$, respectively $\lambda, \mu \in \Gamma_k$ to the energy. Instead of computing the projection of $(\boldsymbol{p}, \boldsymbol{b})$ in each step, we can then perform the projection of the new variables $(\boldsymbol{d}_k, \boldsymbol{b}_k)$ on a corresponding constraint set. The advantage is that the overall projection decouples into independent projections of $\boldsymbol{d}_k^{\lambda}$ or $\boldsymbol{d}_k^{\lambda\mu}$ and b_k^{λ} onto simple convex sets, which are easy to implement. Alternatively, constraints of the form $\big| \boldsymbol{p}_k^{\lambda} - \boldsymbol{p}_k^{\mu} \big|_2 \leq m$ as used in (5.8), (5.11) and (5.18) can be enforced using convex duality, by adding the terms

$$\inf_{\boldsymbol{\eta}} \int_{\Omega} \boldsymbol{\eta}_k^{\lambda,\mu} \big(\boldsymbol{p}_k^{\lambda} - \boldsymbol{p}_k^{\mu} \big) + m \big| \boldsymbol{\eta}_k^{\lambda,\mu} \big|_2 \, \mathrm{d}x \tag{6.11}$$

to the energy instead of (6.10). We used this way in our implementation, as it turns out to be much faster in practice. The optimization (6.7) is now performed over the primals $\boldsymbol{v}$, $\boldsymbol{d}$, $\boldsymbol{\eta}$ and the duals $\boldsymbol{p}$, $\boldsymbol{b}$, $\boldsymbol{q}$, $\boldsymbol{\sigma}$.

Finally, the projection of a $\tilde{\boldsymbol{q}}$ onto $\mathcal{Q}$ consists of solving

$$\operatorname*{argmin}_{\boldsymbol{q} \in \mathcal{Q}} \left\{ \sum_{k=1}^{d} \sum_{\lambda \in \Lambda_k} \big(q_k^{\lambda} - \tilde{q}_k^{\lambda} \big)^2 \right\} \tag{6.12}$$

pointwise for each $x \in \Omega$. The number of constraints in $\mathcal{Q}$, as defined in (4.11), equals the total number of labels in the product space $\Gamma = \Lambda_1 \times \cdots \times \Lambda_d$. Unfortunately, implementing these constraints by adding Lagrange multiplier terms

$$\inf_{\boldsymbol{\mu} \geq 0} - \int_{\Omega} \sum_{\gamma \in \Gamma} \mu^{\gamma}(x) \Big(q_1^{\gamma_1}(x) + \ldots + q_d^{\gamma_d}(x) - c^{\gamma}(x) \Big) \, \mathrm{d}x \tag{6.13}$$

to the *global* problem (6.7), i.e. for each $x \in \Omega$, is not possible for larger problems since it requires too many additional variables $\boldsymbol{\mu}$ to be memory efficient.

Thus, for larger problems, the projection needs to be computed explicitly after each outer iteration as a *subproblem* by solving (6.12), which increases the run time, see Figure 6.1. To make sure that $\boldsymbol{q}$ lies in $\mathcal{Q}$ we add the corresponding Lagrange multiplier terms to the *local* energy (6.12). This results in another saddle point problem to be optimized over now unconstrained $\boldsymbol{q}$ and $\boldsymbol{\mu} \geq 0$, which we again solve using the algorithm in [10]. Specifically, since the $\boldsymbol{q}$-only terms are uniformly convex, we use the "Algorithm 2" of [10] with the accelerated $\mathcal{O}(\frac{1}{N_{\text{iter}}^2})$ convergence rate. Since there is only a small change in the variables $\boldsymbol{q}$ per outer iteration, only a small number of inner iterations is required. In our experiments, we used 10 inner iterations.

6.3. Memory requirements. When the domain Ω is discretized into P pixels, the primal and dual variables are represented as matrices. There are essentialy two types of data. The first type is relatively cheap to store, since memory requirements scale with the sum $N_1 + \ldots + N_d$ of the independent dimensions. The second type is expensive, since it scales with $N_1 \cdots N_d$. The variables and constants appearing in the energy are classified in the following table, where $n = \dim \Omega$ is the dimension of the image domain Ω.

Variable or constant	Floating point numbers
$\boldsymbol{\sigma}$	$P \cdot d$
$\boldsymbol{v}$, $\boldsymbol{q}$	$P \cdot (N_1 + \ldots + N_d)$
$\boldsymbol{p}$	$nP \cdot (N_1 + \ldots + N_d)$
$\boldsymbol{\mu}$, $\boldsymbol{c}$	$P \cdot (N_1 \cdots N_d)$

The relaxation of the regularizer incurs additional costs per label space dimension, depending on the type of regularizer. This is summarized in the following table.

Regularizer	Additional variables	Additional floating point numbers
Potts using (5.3)	–	–
TV or cyclic TV	$\boldsymbol{\eta}_k^\lambda$, $\boldsymbol{d}_k^\lambda$	$2nP \cdot N_k$
Huber TV	$\boldsymbol{\eta}_k^\lambda$, $\boldsymbol{d}_k^\lambda$, b_k^λ	$(2n+1)P \cdot N_k$
Truncated linear	$\boldsymbol{\eta}_k^{\lambda,\mu}$	$nP \cdot N_k(N_k-1)/2$
Potts using (5.18), $\alpha = 0$	$\boldsymbol{\eta}_k^{\lambda,\mu}$	$nP \cdot N_k(N_k-1)/2$
Piecewise smooth	$\boldsymbol{\eta}_k^{\lambda,\mu}$, $\boldsymbol{\eta}_k^\lambda$, $\boldsymbol{d}_k^\lambda$, b_k^λ	$nP \cdot N_k(N_k-1)/2 + (2n+1)P \cdot N_k$

Obviously, truncated linear and piecewise smooth regularization can require a lot of additional memory if the dimensions of the factors are large. They seem therefore practically feasible only when the label space consists of many small factors. However, the projections onto the regularizer constraints can also be solved locally in each iteration for each dimension separately. This removes the need to store these variables globally at the cost of additional computation time.

The most expensive variable is $\boldsymbol{\mu}$, appearing in the global problem (6.13) or local problem (6.12), respectively. If we have enough memory to store it, it is more efficient to solve the global problem. However, it is also possible here to trade off computation time for a reduction in memory requirements by solving the local problem (6.12) in each iteration. For this, note that (6.12) can be separated into *independent* subproblems for each $x \in \Omega$ and can be solved in chunks of points x in parallel. The size of the chunks can be chosen to fit into the available memory, ideally we choose it as large as possible for maximum parallelization.

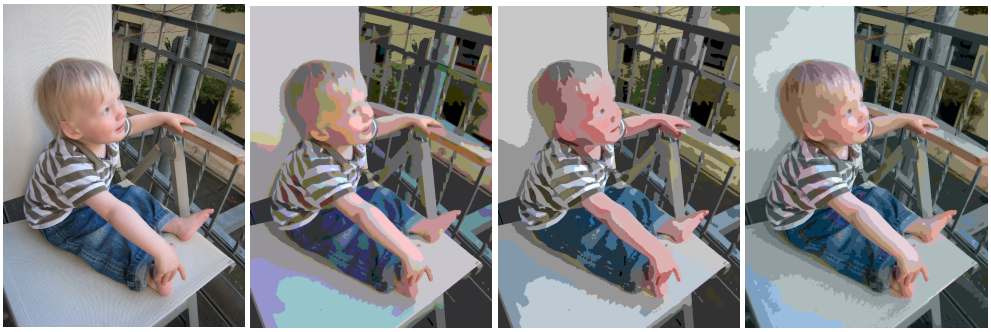
# of Pixels	# Labels	**Memory [Mb]**		**Run time [s]**	
$P = P_x \times P_y$	$N_1 \times N_2$	Previous	Proposed (g/p)	Previous	Proposed (g/p)
320×240	8×8	112	112 / 102	31	26 / 140
320×240	16×16	450	337 / 168	125	80 / 488
320×240	32×32	1800	1124 / 330	503	215 / 1953
320×240	50×50	4394	2548 / 504	2110	950 / 5188
320×240	64×64	7200	4050 / 657	-	1100 / 8090
640×480	8×8	448	521 / 413	127	102 / 560
640×480	16×16	1800	1351 / 676	539	295 / 1945
640×480	32×32	7200	4502 / 1327	-	1290 / 7795
640×480	50×50	17578	10197 / 2017	-	- / 32887
640×480	64×64	28800	16202 / 2627	-	- / 48583

Fig. 6.1: *The table shows the total amount of memory required for the implementations of the previous and proposed methods depending on the size of the problem (using TV regularization). For the proposed method, the projection* (6.12) *of the data term dual variables can be implemented either globally (g), or slower but more memory efficient as a sub-problem of the proximation operator (p), here using* $N_1/5$ *chunks. Also shown is the total run time for 5000 iterations, which usually suffices for convergence. Numbers in red indicate a memory requirement larger than what fits on the largest currently available CUDA capable devices (6 GB). Note that the proposed framework can still handle all problem sizes above.*

Finally, note that the data term $\boldsymbol{c}$ is an expensive constant to store. If there is not enough GPU memory for it, it is possible to e.g. hold it in main memory and transfer separate layers of it during computation of the primal prox operator to the GPU. This increases computation time by a factor of 5-10, so it is usually much more efficient to compute the data term on the fly on the GPU if it is of a simple form.

In summary, with the above reduction techniques it is possible to get rid of all memory expensive variables and constants at the cost of more computation time. To give an idea about the final requirements, they are compared for the case of a 2D label space and TV penalization in Figure 6.1. Note that the memory requirements for the original method without using the reduction are $(n+2)P \cdot (N_1 \cdots N_d)$ to store all primal and dual variables even for the most simple regularizer, while all regularizer costs scale with $N = N_1 \cdots N_d$ according to the table on the previous page. Statistics for a 3D label space with different regularizers can be found in Figure 7.3. Clearly, large scale problems can only be solved using the proposed reduction technique.

7. Experiments. We demonstrate the correctness and usability of our method on several examples. Different regularizers are used in the examples. In the cases where the regularizer can be simulated with the previous relaxation [14], we compared the resulting optimality bounds. On average, our bounds were approximately three times better ($3-5\%$ with the proposed framework compared to $10-15\%$ with the previous relaxation). All experiments were performed with a parallel CUDA implementation running on a NVIDIA GTX 680 GPU for section 7.1, respectively on a TESLA C2070 for all other experiments. The time steps for the algorithm are chosen automatically using the recent preconditioned version [24] of [10]. The number of iterations in each experiment is chosen appropriately so that visually the solution remains stable and does not change anymore (usually 1000-5000 depending on problem size).



Input $RGB\ 6 \times 6 \times 6$ $HSL\ 8 \times 5 \times 5$ $L^*a^*b^*\ 8 \times 5 \times 5$

Fig. 7.1: *Segmentation using different three-dimensional spaces of equidistant color labels. The perceptually uniform color space CIELAB gives the visually most compelling results with the most natural looking color reproduction. Linear penalizer in all channels, except for the H channel, which requires cyclic regularization. Less than two minutes run-time and less than 3% from global optimum for all results.*

7.1. Segmentation. Multi-dimensional label spaces occur naturally in the problem of image segmentation, where a multi-channel input image f is segmented according to a local cost equal to the squared distance to the labels,

$$c^\gamma(x) = \sum_{k=1}^{d} (f_k(x) - \gamma_k)^2. \tag{7.1}$$

Typical multi-channel images are of course color images with various color spaces which are usually three-dimensional, see Figure 7.1 for some segmentation examples.

We choose this archetypical problem for an extensive comparison of our method to different labeling approaches. We first compare our proposed scheme for vectorial multi-label problems (VML) to the similar continuous method for a scalar label space (SML) [21]. For comparisons with discrete approaches, we used the MRF software accompanying the comparative study in [38][2]. We compare to the α-expansion (α-EXP) and α-β-swap (SWAP) algorithms based on the maxflow library [6, 5, 20] using the newest updated implementation for [12][3]. We also compare to two message-passing methods, max-product belief propagation (BP) [39] and sequential tree-reweighted message passing (TRW-S) [40, 18].

Table 7.3 shows detailed statistics of memory requirements, run time and error bounds. Throughout all experiments, we used a fixed number of iterations to keep results comparable, tuned to be sufficient for convergence on intermediate-sized label spaces. For VML and SML, we used 1000 iterations, 50 iterations for TRW-S and BP, and 5 iterations for α-EXP and SWAP.

Memory requirements and largest problem size. Our method can deal with much larger problems than any of the other algorithms. On a high-end workstation with 64 GiB of main memory, the generic implementation of TRW-S already stops working at 12^3 labels for the Potts model. However, the implementation is extremely general and requirements could probably be reduced by more specialized code. About the largest problem which can be solved with α-EXP has 15^3 labels, after which the

[2] MRF 2.1, `http://vision.middlebury.edu/MRF/code/`
[3] gco-v3 library, `http://vision.csd.uwo.ca/code/`

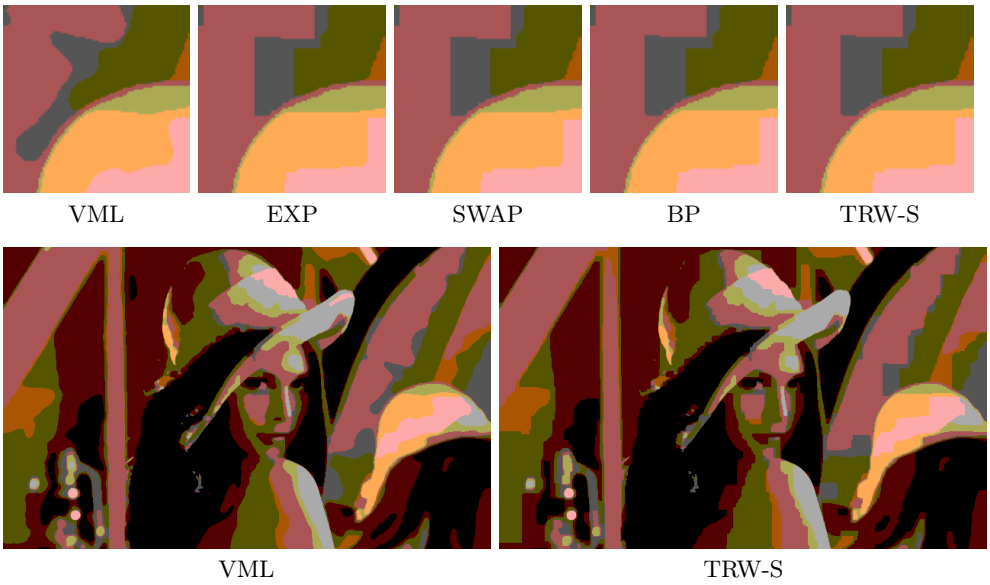


Fig. 7.2: *Closeups of* $4 \times 4 \times 4$ *RGB label segmentations (top) using different algorithms reveal a typical problem of discrete approaches: they exhibit a preference for horizontal and vertical region boundaries since they penalize an* L^1*-distance instead of the correct Euclidean length [17, 42]. The error can be reduced by increasing the neighbourhood connectivity, but only at extremely high costs of memory and computation time. Overall, these metrication errors lead to blocky and visually less pleasing segmentation results (bottom).*

reference implementation segfaults - this hints at a hidden limitation of the implementation, memory-wise it should be able to cope with about 17^3.

In contrast, the GPU has only 4 GiB of memory, and our method can still handle problems up to 31^3 (almost 30000 labels), albeit with high run-time requirements. The table below shows limit cases at an image resolution of 640×950—note that GPU memory shown is the theoretical minimum amount required, but we leverage all the remaining GPU memory to minimize the number of chunks for the local projections.

	VML		α**-EXP**	
label space	GPU mem [MiB]	runtime [min]	CPU mem [MiB]	runtime [min]
$15 \times 15 \times 15$	550	79	18120	31
$23 \times 23 \times 23$	836	485	$>$ 64 GiB	(est. 480)
$31 \times 31 \times 31$	1123	1179	$>$ 64 GiB	(est. 7680)

Performance. For smaller problems when we can use the global implementation of the constraints, our method outperforms the others in terms of run time, while attaining comparable optimality bounds. When the problems become larger, we need to switch to local projections per iteration, which increases run time five-fold and makes the other methods faster across a certain range of problem sizes. However, note that the run-time of our algorithm scales better, so that at problem size 23^3 the algorithm is about to break even with the estimated computation times of α-EXP again, the latter approximately doubling its computation time every time the dimension of each factor is increased by 2, see Figure 7.3.

Potts regularizer (using (5.3) for the continuous methods)

Algorithm	4 × 4 × 4 mem [MiB][1]	time [s][2]	bound [%][3]	6 × 6 × 6 mem [MiB][1]	time [s][2]	bound [%][3]
VML	367	15.25	0.46	998	42.83	0.72
SML	607	16.81	1.43	> 4 GiB	–	–
α-EXP	677	25.13	0.20[4]	1369	88.90	0.54[4]
SWAP	677	29.84	0.28[5]	1369	137.57	0.81[5]
BP	1368	36.63	6.94[5]	4256	119.43	11.56[5]
TRW-S	1368	40.06	0.16	4256	129.52	0.80
	8 × 8 × 8			**10 × 10 × 10**		
	mem [MiB][1]	time [s][2]	bound [%][3]	mem [MiB][1]	time [s][2]	bound [%][3]
VML	2173	94.59	1.03	2185[6]	209.34[6]	0.80
α-EXP	2746	173.64	0.90[4]	5020	343.67	1.01[4]
SWAP	2746	461.48	1.34[5]	5020	1472.45	1.62[5]
BP	8667	254.08	16.29[5]	16610	496.12	17.73[5]
TRW-S	8667	287.30	1.95	16610	539.51	2.70

Linear (TV) regularizer

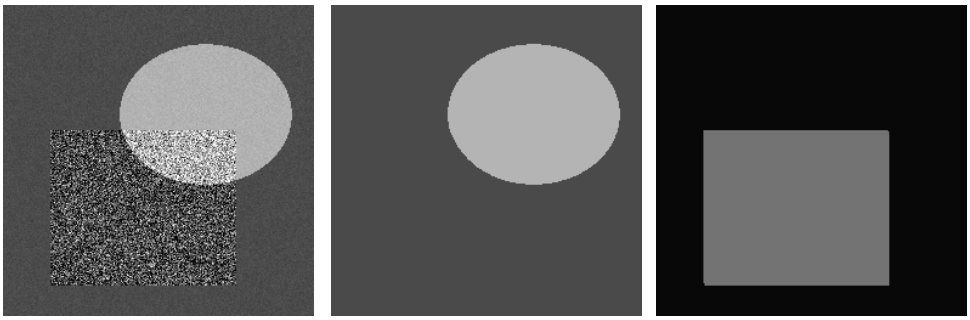
Algorithm	4 × 4 × 4 mem [MiB][1]	time [s][2]	bound [%][3]	6 × 6 × 6 mem [MiB][1]	time [s][2]	bound [%][3]
VML	401	18.67	1.50	1055	48.09	1.67
α-EXP	677	18.04	0.06[4]	1369	82.70	–[4]
SWAP	677	23.67	0.16[5]	1369	139.90	–[5]
BP	51000[8]	740.49[8]	5.87[5]	> 64 GiB	–	–
TRW-S	51000[8]	746.00[8]	0.04	> 64 GiB	–	–
	8 × 8 × 8			**10 × 10 × 10**		
	mem [MiB][1]	time [s][2]	bound [%][3]	mem [MiB][1]	time [s][2]	bound [%][3]
VML	2253	101.75	2.27	2287[6]	217.66[6]	3.10
α-EXP	2746	201.05	–[4]	5020	408.93	–[4]
SWAP	2746	504.47	–[5]	5020	1576.32	–[5]

Truncated linear regularizer

Algorithm	4 × 4 × 4 mem [MiB][1]	time [s][2]	bound [%][3]	6 × 6 × 6 mem [MiB][1]	time [s][2]	bound [%][3]
VML	435	20.77	0.95	1168	55.01	1.83
α-EXP	677	18.13	0.09[4]	1369	82.78	–[4]
SWAP	677	23.68	0.18[5]	1369	139.30	–[5]
BP	51000[8]	750.36[8]	4.49[5]	> 64 GiB	–	–
TRW-S	51000[8]	741.80[8]	0.04	> 64 GiB	–	–
	8 × 8 × 8			**10 × 10 × 10**		
	mem [MiB][1]	time [s][2]	bound [%][3]	mem [MiB][1]	time [s][2]	bound [%][3]
VML	1523[6]	132.33[6]	3.24	805[6,7]	1304[6,7]	3.50
α-EXP	2746	200.11	–[4]	5020	408.73	–[4]
SWAP	2746	507	–[5]	5020	1578.29	–[5]

1: GPU memory for VML and SML, otherwise CPU memory (measured with `valgrind` memory profiler).
2: Intel Core i7-3820 CPU @ 3.8 GHz with 64 GB of RAM, nVidia GTX 680 GPU with 4 GB of RAM.
3: Optimality bound in percent of the lower bound to solution provided by the algorithm.
4: No lower bound provided by algorithm, a theoretical (far from tight) a-priori bound is known to be ≥100% [6, 12]. Bound was computed with lower bound returned by TRW-S, if any is given.
5: No lower bound provided by algorithm, no theoretical bound known as far as we know. See also 4.
6: Data term computed on the fly (saves GPU storage, minimal runtime increase).
7: Data term relaxation reprojection computed in each iteration, reduces GPU memory usage by 2 GiB, but increases computation time by a factor of 5.
8: Requires general regularizer implementation, which is inefficient according to the author of the code. Unfortunately, no specialized implementation is available for 3D regularizers.

Fig. 7.3: *Performance comparison of several continuous and discrete multilabel algorithms on the segmentation problem using a 3D label space and different regularizers. Results are averaged over 10 different images with average resolution of* 0.7 *Megapixels. See section 7.1 for a discussion.*



Input, spatially varying noise Denoised image and reconstructed standard deviation

Fig. 7.4: *The algorithm allows to jointly recover the unknown standard deviation σ of the noise as well as the intensity of a denoised image by solving a single optimization problem. Ground truth: Within rectangle Gaussian noise with standard deviation $\sigma = 0.25$, outside $\sigma = 0.02$; image intensity within ellipsoid $u = 0.7$, outside $u = 0.3$. Image resolution is 256×256 using 32×32 labels. Computation time is 4.4 minutes.*

Input, textured object Simultaneous piecewise smooth approximation of intensity (left) and standard deviation (right)

Fig. 7.5: *A piecewise smooth image approximation of both intensity and noise standard deviation using (7.2) and the Mumford-Shah regularizer for both u and σ. This model allows to separate textured objects in a natural way by jointly estimating the mean and standard deviation of image intensities. The amount of smoothing is stronger in region of larger standard deviation. Image resolution is 320×214 using 32×32 labels, leading to a run time of 10.3 minutes.*

From a theoretical point of view, the move-making schemes α-EXP and SWAP solve the problem by iterating binary decisions instead of dealing with the whole problem at once, which explains their efficiency. However, as a result they cannot give reasonable optimality bounds, see remarks in Figure 7.3. As an additional drawback, all discrete methods suffer from metrication errors [17, 42], as detailed in Figure 7.2.

7.2. Adaptive denoising. As a novel application of a multi-dimensional label space, we present adaptive denoising, where we *jointly* estimate a noise level and a denoised image by solving a single minimization problem. Note that here we require the *continuous* label space to represent the image intensity range.

The Mumford-Shah energy can be interpreted as a denoising model which yields the maximum a posteriori estimate for the original image under the assumption that the input image f was distorted with Gaussian noise of standard deviation σ. An interesting generalization of this model is when the standard deviation of the noise is not constant but rather varies over the image. Viewing it as an additional unknown,

Fig. 7.6: *The proposed method can be employed to simultaneously optimize for a displacement and an occlusion map. This problem is also too large to be solved by alternative relaxation methods on current GPUs. From left to right: Left and right input image I_L and I_R, and computed disparity and occlusion map; red areas denote occluded pixels.*

the label space becomes two-dimensional, with one dimension representing the unknown intensity u of the original image, and the second dimension representing the unknown standard deviation σ of the noise. The data term of the energy can then be written as [7]

$$\int_\Omega \frac{(u-f)^2}{2\sigma^2} + \frac{1}{2}\log(2\pi\sigma^2)\,\mathrm{d}x. \tag{7.2}$$

Results of the optimization can be observed in Figure 7.4 and Figure 7.5. For the regularizer, we used piecewise constant Mumford-Shah for both σ and u in Figure 7.4, and piecewise smooth Mumford-Shah in Figure 7.5. In the real world example Figure 7.5, the solution can be interpreted as a uniformly smooth approximation, where all regions attain a similar smoothness level regardless of the amount of texture in the input.

7.3. Depth and Occlusion map. In this test, we simultaneously compute a depth map and an occlusion map for a stereo pair of two color input images $I_L, I_R : \Omega \to \mathbb{R}^3$. The occlusion map shall be a binary map denoting whether a pixel in the left image has a matching pixel in the right image. Thus, the space of labels is two-dimensional with Λ_1 consisting of the disparity values and a binary $\Lambda_2 = \{0, 1\}$ for the occlusion map. We use the TV smoothness penalty on the disparity values. The Potts regularizer is imposed for the occlusion map. The distance on the label space thus becomes

$$d(\gamma, \chi) = s_1 \left|\gamma_1 - \chi_1\right| + s_2 \left|\gamma_2 - \chi_2\right| , \tag{7.3}$$

with suitable weights $s_1, s_2 > 0$. We penalize an occluded pixel with a constant cost $c_{occ} > 0$, which corresponds to a threshold for the similarity measure above which we believe that a pixel is not matched correctly anymore. The cost associated with a label γ at $(x, y) \in \Omega$ is then defined as

$$c^\gamma(x,y) = \begin{cases} c_{occ} & \text{if } \gamma_2 = 1, \\ \left|I_L(x,y) - I_R(x-\gamma_1, y)\right|_2 & \text{otherwise.} \end{cases} \tag{7.4}$$

The result for the "Moebius" test pair from the Middlebury benchmark is shown in Figure 7.6. The input image resolution was scaled to 640×512, requiring 128 disparity labels, which resulted in a total memory consumption which was slightly too big for previous methods, but still in reach of the proposed algorithm. Total computation time required was 1170 seconds.

Dataset		VML			VML-ECCV [14]			TV-L^1 [43]	
size	Δ	aep	aan	bound [%]	aep	aan	bound [%]	aep	aan
Venus									
420 × 380	10	0.39	4.25	0.21	0.81	5.44	5.88	0.44	7.74
Dimetrodon									
584 × 388	5	0.22	4.58	6.12	0.62	6.38	7.87	0.22	3.94
Hydrangea									
584 × 388	12	0.42	4.06	2.64	0.81	5.60	9.26	0.22	2.64
RubberWhale									
584 × 388	5	0.18	5.73	2.36	0.29	6.12	8.60	0.20	6.29
Grove2									
640 × 480	5	0.34	4.54	5.01	0.55	6.16	13.25	0.22	3.12
Grove3									
640 × 480	15	1.06	12.02	9.22	2.01	14.49	10.50	0.76	7.41
Urban2									
640 × 480	22	0.81	9.31	1.07	0.97	8.15	6.32	0.47	3.51
Urban3									
640 × 480	18	1.38	8.95	1.05	1.65	10.82	4.99	0.90	8.02

Fig. 7.7: *Accuracy comparison on Middlebury data sets. Maximum displacement (Δ) and average endpoint error (aep) are measured in pixels, average angular error (aan) in degrees. Not surprisingly, accuracy for the vectorial multi-label (VML) method is strongly correlated with the amount of labels per pixel, and thus decreases with larger maximum displacement. On data sets with small maximum displacement, the accuracy using* 35×35 *labels is comparable to TV-*L^1 *optical flow, while other data sets would require either coarse-to-fine schemes or a greater number of labels for the method to remain competitive. The proposed new relaxation outperformes the previous one [14] in all respects.*

7.4. Optic Flow. In this experiment, we compute optic flow between two color input images $I_0, I_1 : \Omega \to \mathbb{R}^3$ taken at two different time instants. The space of labels is again two-dimensional, with $\Lambda_1 = \Lambda_2$ denoting the possible components of flow vectors in x and y-direction, respectively. We regularize both directions with either TV or a truncated linear penalty on the component distance, i.e.

$$d(\gamma, \chi) = s \min(t, |\gamma_1 - \chi_1|) + s \min(t, |\gamma_2 - \chi_2|)\,, \tag{7.5}$$

with a suitable manually chosen weight $s > 0$ and threshold $t > 0$. Note that we can provide a tight relaxation of the exact penalizer, which was only coarsely approximated in the previous approaches [14, 21]. The cost function just compares pointwise pixel colors in the images, i.e.

$$c^\gamma(x, y) = |I_0(x, y) - I_1(x + \gamma_1, y + \gamma_2)|_2\ . \tag{7.6}$$

Results can be observed in Figures 1.1, 7.8, 7.9 and 8.1. See Figure 7.9 for the color code of the flow vectors. In all examples, the number of labels is so high that this problem is currently impossible to solve with previous convex relaxation techniques by a large margin, see Figure 6.1.

Compared to the relaxation proposed in the original conference publication [14], total computation time was reduced dramatically, see Figure 7.8. The large computation time for the TV relaxation of [14] is caused by an overly restrictive constraint on the time steps due to the structure of the embedding matrix A_k in (5.3). Using $N_1 = N_2 := N$ labels in each direction of the 2-dimensional optic flow label space, the time steps can be seen to be proportional to $N^{-3/2}$. In contrast, the proposed TV relaxation in Section 5.3 allows larger time steps proportional to $N^{-1/2}$ and thus leads to a substantially lower number of iterations. We suggest to use the preconditioning variant of the algorithm [10] where the time steps are chosen adaptively.

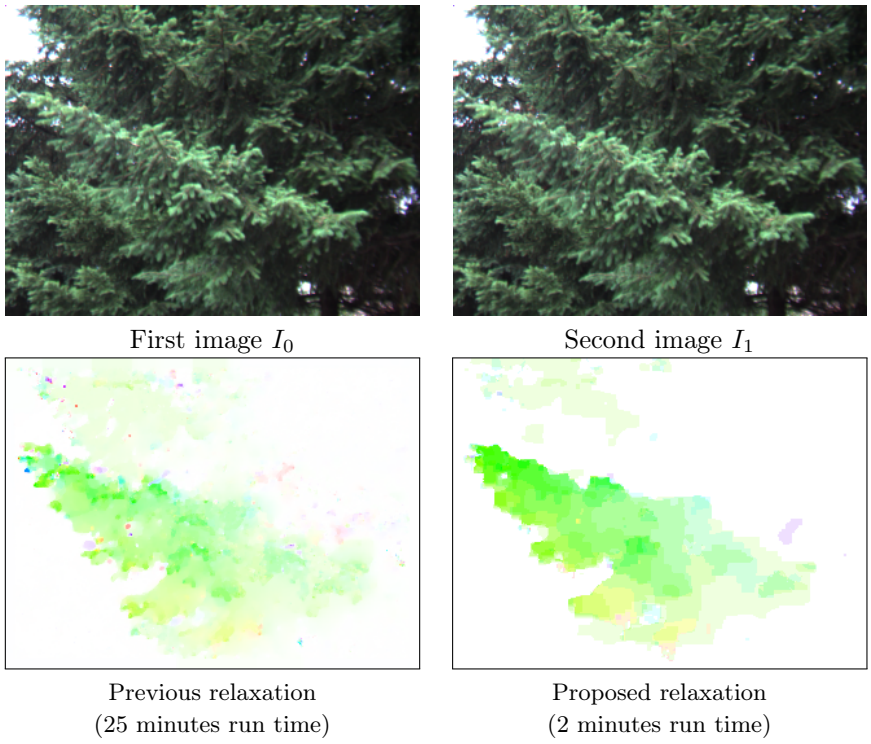
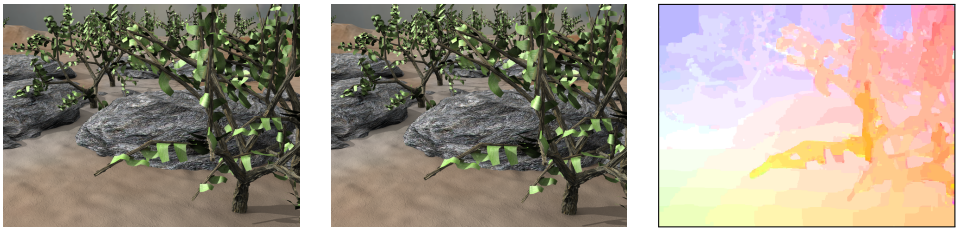


Fig. 7.8: *Optical flow fields with* 32×32 *labels computed on an image with resolution* 320×240 *using TV regularization. With the new relaxation of the regularizers, we achieve optimality bounds which are on average three times lower than with previous relaxations from [14, 21], using the proposed data term relaxation* (4.10) *for both cases. The result in the lower left is computed with the TV relaxation from [21]. Since the scaling of the regularity term is not directly comparable, we chose optimal parameters for both algorithms manually. The large time difference results from a narrow constraint on the time step, see Section 7.4.*

Due to the global optimization of a convex energy, we can successfully capture large displacements without having to implement a coarse-to-fine scheme, see Figure 7.9. Figure 7.7 shows detailed numeric results of our method on the data sets of the Middlebury benchmark with public ground truth available. We compare our current method with a linear regularizer using 35×35 labels to our old relaxation [14] and TV-L^1 optic flow [43], which utilizes a very similar energy which is optimized with a coarse-to-fine scheme and quadratic relaxation of the linearized functional. Results show that we get reasonable optimality bounds for the energy and are in most cases within 5% of the global optimum, while accuracy of the actual optical flow results depends on how fine the discretization is compared to the maximum displacement. The new method is obviously superior to the old relaxation in all respects—the previous one is only provided for reference, and we strongly recommend to use the new one proposed in this work. For a method which is competitive on the Middlebury benchmark, we would need to further increase the amount of labels by e.g. implementing a coarse-to-fine scheme, and fine-tune our data terms. This is however out of the scope of this paper, since our focus is to provide an efficient optimization framework.

First image I_0 Second image I_1 Flow field and color code

Fig. 7.9: *When employed for optic flow, the proposed method can successfully capture large displacements without the need for coarse-to-fine approaches, since a global optimization is performed over all labels. In contrast to existing methods, our solution is within a known bound of the global optimum.*



First image I_0 Second image I_1 Flow field

Fig. 8.1: *Example with a larger image resolution of* 640×480 *pixels, which requires* 32×32 *labels. Regularizer is the total variation in each component. Computation time is* 21.6 *minutes.*

8. Conclusion. We have introduced a continuous convex relaxation for multi-label problems where the label space has a product structure and the regularizer is separable. Such labeling problems are plentiful in computer vision. The proposed reduction method improves on previous methods in that it requires orders of magnitude less memory and computation time, while retaining the advantages: a very flexible choice of regularizer on the label space, a globally optimal solution of the relaxed problem and an efficient parallel GPU implementation with guaranteed convergence.

The proposed framework combines the advantages of the efficient multi-dimensional data term relaxation [36] with the tight relaxation of the regularizers in [9]. It allows for a very general class of continuous regularizers on multi-dimensional label spaces and can thus solve a significant range of problems efficiently. For example, we can explicitly encourage the solution to be smooth in certain regions, and can represent Huber-TV and truncated linear regularization by an exact and tight relaxation. The regularizers can be arbitrarily mixed, in the sense that each dimension of the label space can have its own type of regularity. Because of the reduced memory requirements, we can successfully handle specific problems with very large number of labels, which could not be done with previous labeling methods. A systematic experimental comparison with respective discrete algorithms (α-EXP, SWAP, BP, TRW-S) shows a good performance and often improved results.

Acknowledgements. We thank Antonin Chambolle (École Polytechnique) for helpful discussions concerning the proof of Proposition 4.4. This work was supported by the ERC Starting Grant "Convex Vision".

REFERENCES

[1] G. Alberti, G. Bouchitté, and G. Dal Maso, *The calibration method for the Mumford-Shah functional*, C. R. Acad. Sci. Paris Sér. I Math., 329 (1999), pp. 249–254. 4

[2] ———, *The calibration method for the Mumford-Shah functional and free-discontinuity problems*, Calculus of Variations and Partial Differential Equations, 16 (2003), pp. 299–333. 4, 6, 7, 9, 16, 32

[3] H. Attouch, G. Buttazzo, and G. Michaille, *Variational Analysis in Sobolev and BV Spaces*, MPS-SIAM Series on Optimization, Society for Industrial and Applied Mathematics, 2006. 7, 8, 37

[4] Y. Boykov and V. Kolmogorov, *Computing geodesics and minimal surfaces via graph cuts*, in IEEE International Conference on Computer Vision (ICCV), 2003, pp. 26–33. 3

[5] Y. Boykov and V. Kolmogorov, *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004). 22

[6] Y. Boykov, O. Veksler, and R. Zabih, *Fast approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23 (2001), pp. 1222–1239. 3, 4, 22, 24

[7] T. Brox and D. Cremers, *On local region models and a statistical interpretation of the piecewise smooth Mumford-Shah functional*, International Journal of Computer Vision, 84 (2009), pp. 184–193. 26

[8] A. Chambolle, *Finite-differences discretizations of the Mumford-Shah functional*, Mathematical Modelling and Numerical Analysis, 112 (1999), pp. 261–288. 7

[9] A. Chambolle, D. Cremers, and T. Pock, *A convex approach for computing minimal partitions*, Tech. Report TR-2008-05, Dept. of Computer Science, University of Bonn, 2008. 1, 4, 6, 12, 15, 17, 29

[10] A. Chambolle and T. Pock, *A first-order primal-dual algorithm for convex problems with applications to imaging*, Journal of Mathematical Imaging and Vision, 40 (2011), pp. 120–145. 18, 20, 21, 27

[11] D. Cremers and K. Kolev, *Multiview stereo and silhouette consistency via convex functionals over convex domains*, IEEE Transactions on Pattern Analysis and Machine Intelligence, (2010). 1

[12] A. Delong, A. Osokin, H. Isack, and Y. Boykov, *Fast approximate energy minimization with label costs*, International Journal of Computer Vision, 96 (2012), pp. 1–27. 22, 24

[13] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas, and N. Navab, *Optical flow estimation with uncertainties through dynamic MRFs*, in Proc. International Conference on Computer Vision and Pattern Recognition, 2008. 3

[14] B. Goldluecke and D. Cremers, *Convex relaxation for multilabel problems with product label spaces*, in Proc. European Conference on Computer Vision, 2010. 1, 3, 4, 9, 10, 11, 12, 13, 15, 21, 27, 28

[15] D. Greig, B. Porteous, and A. Seheult, *Exact maximum a posteriori estimation for binary images*, J. Royal Statistics Soc., 51 (1989), pp. 271–279. 3

[16] H. Ishikawa, *Exact optimization for Markov random fields with convex priors*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 25 (2003), pp. 1333–1336. 3, 4

[17] M. Klodt, T. Schoenemann, K. Kolev, M. Schikora, and D. Cremers, *An experimental comparison of discrete and continuous shape optimization methods*, in European Conference on Computer Vision (ECCV), October 2008, pp. 332–345. 4, 23, 25

[18] V. Kolmogorov, *Convergent tree-reweighted message passing for energy minimization*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 28 (2006). 22

[19] V. Kolmogorov and C. Rother, *Minimizing non-submodular functions with graph cuts - a review*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 29 (2007), pp. 1274–1279. 3

[20] V. Kolmogorov and R. Zabih, *What energy functions can be minimized via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26 (2004), pp. 147–159. 3, 22

[21] J. Lellmann, F. Becker, and C. Schnörr, *Convex optimization for multi-class image labeling with a novel family of total variation based regularizers*, in IEEE International Conference on Computer Vision (ICCV), 2009. 1, 4, 7, 12, 13, 14, 15, 22, 27, 28

[22] C. Michelot, *A finite algorithm for finding the projection of a point onto the canonical simplex of $\mathbb{R}^n$*, J. Optimization Theory and Applications, 50 (1986), pp. 195–200. 19

[23] M. Nikolova, S. Esedoglu, and T.Chan, *Algorithms for finding global minimizers of image segmentation and denoising models*, SIAM Journal of Applied Mathematics, 66 (2006), pp. 1632–1648. 1, 4, 18

[24] T. Pock and A. Chambolle, *Diagonal preconditioning for first order primal-dual algorithms in convex optimization*, in International Conference on Computer Vision (ICCV 2011), 2011. 21

[25] T. Pock, A. Chambolle, H. Bischof, and D. Cremers, *A convex relaxation approach for computing minimal partitions*, in IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2009, pp. 810–817. 4, 6, 7, 17

[26] T. Pock, D. Cremers, H. Bischof, and A. Chambolle, *An algorithm for minimizing the piecewise smooth Mumford-Shah functional*, in Proc. International Conference on Computer Vision, 2009. 1, 6, 7, 12, 15, 16

[27] ———, *Global solutions of variational models with convex regularization*, SIAM Journal on Imaging Sciences, (2010). 1, 4, 6, 7, 12, 14, 15, 16

[28] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers, *A convex formulation of continuous multi-label problems*, in European Conference on Computer Vision (ECCV), 2008, pp. 792–805. 4, 7

[29] S. Ramalingam, P. Kohli, K. Alahari, and P. Torr, *Exact inference in multi-label CRFs with higher order cliques*, in Proc. International Conference on Computer Vision and Pattern Recognition, 2008. 3

[30] R. T. Rockafellar, *Convex Analysis*, Princeton University Press, 1996. 34

[31] D. Schlesinger and B. Flach, *Transforming an arbitrary min-sum problem into a binary one*, tech. report, Dresden University of Technology, 2006. 3

[32] A. Shekhovtsov, J. Garcia-Arteaga, and T. Werner, *A discrete search method for multimodal non-rigid image registration*, in Proceedings of the 2008 IEEE CVPR Workshop on Non-Rigid Shape Analysis and Deformable Image Alignment, 2008. 3

[33] A. Shekhovtsov, I. Kovtun, and V. Hlavac, *Efficient MRF deformation model for non-rigid image matching*, Computer Vision and Image Understanding, 112 (2008), pp. 91–99. 3

[34] G. Strang, *Maximal flow through a domain*, Mathematical Programming, 26 (1983), pp. 123–143. 3

[35] E. Strekalovskiy and D. Cremers, *Total variation for cyclic structures: Convex relaxation and efficient minimization*, in Proc. International Conference on Computer Vision and Pattern Recognition, 2011. 3, 15, 16

[36] E. Strekalovskiy, B. Goldluecke, and D. Cremers, *Tight convex relaxations for vector-valued labeling problems*, in Proc. International Conference on Computer Vision, 2011. 3, 29

[37] R. Szeliski, *Bayesian modeling of uncertainty in low-level vision*, International Journal of Computer Vision, 5 (1990), pp. 271–301. 3

[38] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, *A comparative study of energy minimization methods for Markov random fields*, in Proc. European Conference on Computer Vision, 2006. 22

[39] M. Tappen and W. Freeman, *Comparison of graph cuts with belief propagation for stereo, using identical MRF parameters*, in Proc. International Conference on Computer Vision, 2003. 22

[40] M. Wainwright, T. Jaakkola, and A. Willsky, *Map estimation via agreement on trees: message-passing and linear programming*, IEEE Trans. Inf. Theory, 51 (2005), pp. 3697–3717. 3, 22

[41] C. Zach, D. Gallup, J. Frahm, and M. Niethammer, *Fast global labeling for real-time stereo using multiple plane sweeps*, in Vision, Modeling and Visualization, 2009, pp. 243–252. 1, 4

[42] C. Zach, C. Haene, and M. Pollefeys, *What is optimized in tight convex relaxations for multi-label problems?*, in Proc. International Conference on Computer Vision and Pattern Recognition, 2012. 23, 25

[43] C. Zach, T. Pock, and H. Bischof, *A duality based approach for realtime $TV-L^1$ optical flow*, in Pattern Recognition (Proc. DAGM), 2007, pp. 214–223. 14, 27, 28

Appendix: proofs of propositions and theorems in the main paper.

8.1. Proof of proposition 3.1. In order to proof the proposition, we show that the mapping induces a pointwise bijection from $\Delta_\times$ onto Δ. We first show it is onto: for $\boldsymbol{u}(x)$ in Δ, there exists exactly one $\gamma \in \Gamma$ with $u^\gamma(x) = 1$. Set $v_k^\lambda(x) = 1$ if $\lambda = \gamma_k$, and $v_k^\lambda(x) = 0$ otherwise. Then equation (3.5) is satisfied as desired, see Figure 3.1. To see that the map is one-to-one, we just count the elements in $\Delta_\times$. Since Δ_k contains N_k elements, the number of elements in $\Delta_\times$ is $N_1 \cdot \ldots \cdot N_d = N$, the same as in Δ. ∎

8.2. Proof of theorem 4.1. Since derivatives of indicator functions do not exist in ordinary sense, the integral in (4.1) is meant to be a convenient notation for

$$\int_{\Omega \times \Lambda_k} (\boldsymbol{\phi}^1, \phi^2) \cdot \nu_{\Gamma_{u_k}} \, \mathrm{d}\mathcal{H}^n(x, \lambda) \tag{8.1}$$

where

$$\Gamma_{u_k} := \Big\{ (x, u(x)) \,\Big|\, x \in \Omega \setminus S_{u_k} \Big\} \cup \Big\{ (x, s) \,\Big|\, x \in S_{u_k},\ s \in [u_k^-, u_k^+] \Big\} \tag{8.2}$$

is the extended graph of u, and $\nu_{\Gamma_{u_k}}$ is the normal on Γ_{u_k} pointing "downwards". Intuitively, $\nabla 1_{\mathrm{hyp}(u_k)}$ in (4.1) is nonzero only on Γ_{u_k}, and equals $\nu_{\Gamma_{u_k}}$ up to a delta function factor. For a fixed $\boldsymbol{\phi}$ denote the integral (8.1) by $J_{\boldsymbol{\phi}}$. It is equal to [2, lemma 2.8]

$$\begin{aligned} J_{\boldsymbol{\phi}} = &\int_{\Omega \setminus S_{u_k}} \Big(\boldsymbol{\phi}^1(x, u_k) \cdot \nabla u_k - \phi^2(x, u_k) \Big) \, \mathrm{d}x \\ &+ \int_{S_{u_k}} \left(\int_{u_k^-}^{u_k^+} \boldsymbol{\phi}^1(x, s) \, \mathrm{d}s \right) \cdot \nu_{u_k} \, \mathrm{d}\mathcal{H}^{n-1}(x). \end{aligned} \tag{8.3}$$

Define $\boldsymbol{p} : \Omega \times \Lambda_k \to \mathbb{R}^n$ and $b : \Omega \times \Lambda_k \to \mathbb{R}$ by

$$\boldsymbol{p}(x, \lambda) := \int_{\lambda_0}^{\lambda} \boldsymbol{\phi}^1(x, s) \, \mathrm{d}s, \qquad b(x, \lambda) := \phi^2(x, \lambda) \tag{8.4}$$

for some $\lambda_0 \in \Lambda_k$. Since $\phi \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R})$, also $(p, b) \in C_c^1(\Omega \times \Lambda_k; \mathbb{R}^n \times \mathbb{R})$. With these new variables we have

$$\begin{aligned} J_{\boldsymbol{\phi}} = &\int_{\Omega \setminus S_{u_k}} \Big(\partial_\lambda \boldsymbol{p}(x, u_k) \cdot \nabla u_k - b(x, u_k) \Big) \, \mathrm{d}x \\ &+ \int_{S_{u_k}} \Big(\boldsymbol{p}(x, u_k^+) - \boldsymbol{p}(x, u_k^-) \Big) \cdot \nu_{u_k} \, \mathrm{d}\mathcal{H}^{n-1}(x). \end{aligned} \tag{8.5}$$

By the divergence theorem,

$$\begin{aligned} \int_{\Omega \setminus S_{u_k}} \mathrm{div}\big(\boldsymbol{p}(x, u_k)\big) \, \mathrm{d}x = &\int_{S_{u_k}} \Big(\boldsymbol{p}(x, u_k^+) \cdot (-\nu_{u_k}) + \boldsymbol{p}(x, u_k^-) \cdot \nu_{u_k} \Big) \, \mathrm{d}\mathcal{H}^{n-1}(x) \\ &+ \int_{\partial\Omega} \boldsymbol{p}(x, u_k) \cdot \nu_{\partial\Omega} \, \mathrm{d}\mathcal{H}^{n-1}(x). \end{aligned} \tag{8.6}$$

In the integrand of the first integral on the right hand side there are two addends for each point of S_{u_k}, because the integration on the left hand side is performed on *both*

sides of S_{u_k}. The outer normal for the u_k^- side is ν_{u_k} by definition, and for the u_k^+ side it is just the opposite. The last integral on the right hand side is zero because $\boldsymbol{\phi}$ and therefore also $\boldsymbol{p}$ has compact support in Ω. Using (8.6) in (8.5) we obtain

$$J_{\boldsymbol{\phi}} = \int_{\Omega\setminus S_{u_k}} \Big(\partial_\lambda \boldsymbol{p}(x,u_k)\cdot\nabla u_k - b(x,u_k)\Big)\,\mathrm{d}x - \int_{\Omega\setminus S_{u_k}} \mathrm{div}\big(\boldsymbol{p}(x,u_k)\big)\,\mathrm{d}x \tag{8.7}$$

By the chain rule,

$$\mathrm{div}\big(\boldsymbol{p}(x,u_k)\big) = (\mathrm{div}\boldsymbol{p})(x,u_k) + \partial_\lambda \boldsymbol{p}(x,u_k)\cdot\nabla u_k. \tag{8.8}$$

Thus, the expression (8.7) simplifies to

$$J_{\boldsymbol{\phi}} = \int_{\Omega\setminus S_{u_k}} \big(-(\mathrm{div}\boldsymbol{p})(x,u_k) - b(x,u_k)\big)\,\mathrm{d}x = \int_{\Omega\times\Lambda_k} (-\mathrm{div}\boldsymbol{p} - b)\,\boldsymbol{v}_k\;\mathrm{d}(x,\lambda). \tag{8.9}$$

The last equality is simply the definition of how the distribution $\boldsymbol{v}_k(x,\lambda)=\delta(u_k-\lambda)$, defined for $u_k\in\mathcal{SBV}(\Omega)$, acts on functions. Now, the claim of the proposition follows directly from (4.1) and (8.9). □

8.3. Proof of proposition 4.2. Since at each point $x\in\Omega$, $\gamma(x)$ is the label indicated by $\boldsymbol{v}(x)$, by the defining property (3.8) we have $v_k^\lambda(x)=1$ for $\lambda=\gamma_k(x)$ and $v_k^\lambda(x)=0$ for all $\lambda\in\Lambda_k$ with $\lambda\neq\gamma_k(x)$. Thus, for all $\boldsymbol{q}\in\mathcal{Q}$,

$$\sum_{k=1}^{d}\sum_{\lambda\in\Lambda_k} q_k^\lambda(x)v_k^\lambda(x) = \sum_{k=1}^{d} q_k^{\gamma_k(x)} \le c^{\gamma(x)}(x). \tag{8.10}$$

This shows that at least $R_{\mathrm{data}}(\boldsymbol{v}) \le \int_\Omega c^{\gamma(x)}(x)\,\mathrm{d}x = E_{\mathrm{data}}(\boldsymbol{v})$. To prove equality, first observe that we can safely interchange the supremum over $\boldsymbol{q}\in\mathcal{Q}$ with the integration over Ω since the constraints in $\mathcal{Q}$ on $\boldsymbol{q}$ are pointwise in x. This means that we only need to show the pointwise equality, i.e. that for each fixed $x\in\Omega$ the integrand in (4.10) yields $c^\gamma(x)$ when taking the supremum over $\boldsymbol{q}\in\mathcal{Q}$. We use Lagrange multipliers $\boldsymbol{\mu}$ to write the constraints in (4.11) as additional energy terms:

$$\begin{aligned} R_{\mathrm{data}}(\boldsymbol{v}) &= \sup_{\boldsymbol{q}\in\mathcal{Q}} \sum_{k=1}^{d}\sum_{\lambda\in\Lambda_k} q_k^\lambda v_k^\lambda \\ &= \sup_{\boldsymbol{q}} \inf_{\mu^{\widehat{\gamma}}\ge 0} \sum_{k=1}^{d}\sum_{\lambda\in\Lambda_k} q_k^\lambda v_k^\lambda \;-\; \sum_{\widehat{\gamma}\in\Gamma} \mu^{\widehat{\gamma}}\big(q_1^{\widehat{\gamma}_1} + \ldots + q_d^{\widehat{\gamma}_d} - c^{\widehat{\gamma}}\big) \\ &= \inf_{\mu^{\widehat{\gamma}}\ge 0} \sum_{\widehat{\gamma}\in\Gamma} \mu^{\widehat{\gamma}} c^{\widehat{\gamma}} \;+\; \sup_{\boldsymbol{q}} \sum_{k=1}^{d}\sum_{\lambda\in\Lambda_k} q_k^\lambda \Big(v_k^\lambda - \sum_{\widehat{\gamma}\in\Gamma:\,\widehat{\gamma}_k=\lambda} \mu^{\widehat{\gamma}}\Big), \end{aligned} \tag{8.11}$$

interchanging the ordering of $\sup_{\boldsymbol{q}}$ and $\inf_{\boldsymbol{\mu}}$. Evaluating the supremum over $\boldsymbol{q}$ leads to constraints on the variables $\mu^{\widehat{\gamma}}$ and we obtain

$$R_{\mathrm{data}}(\boldsymbol{v}) = \inf_{\mu^{\widehat{\gamma}}\ge 0} \sum_{\widehat{\gamma}\in\Gamma} \mu^{\widehat{\gamma}} c^{\widehat{\gamma}} \tag{8.12}$$

with $\mu^{\widehat{\gamma}}$ such that additionally

$$\sum_{\widehat{\gamma}\in\Gamma:\,\widehat{\gamma}_k=\lambda} \mu^{\widehat{\gamma}} = v_k^\lambda \quad \text{for all } 1\le k\le d \text{ and } \lambda\in\Lambda_k. \tag{8.13}$$

First, for any fixed $1 \leq k \leq d$ and any $\lambda \in \Lambda_k$ with $\lambda \neq \gamma_k$, by assumption we have $v_k^\lambda = 0$. Since $\mu^{\widehat{\gamma}} \geq 0$, (8.13) then gives $\mu^{\widehat{\gamma}} = 0$ for all $\widehat{\gamma} \in \Gamma$ with $\widehat{\gamma}_k \neq \gamma_k$. Combining this for all $1 \leq k \leq d$ we get $\mu^{\widehat{\gamma}} = 0$ for all $\widehat{\gamma} \neq \gamma$. Next, plug $\lambda = \gamma_k$ for some k into (8.13). Since any other addend $\mu^{\widehat{\gamma}}$ is zero, the sum is just μ^γ, while the right hand side is $v_k^{\gamma_k} = 1$.

Therefore, the constraints (8.13) ensure that $\mu^{\widehat{\gamma}} = 0$ for all $\widehat{\gamma} \neq \gamma$ and $\mu^\gamma = 1$, so that (8.12) gives $R_{\text{data}}(\boldsymbol{v}) = c^\gamma$. ∎

8.4. Proof of theorem 4.3. Let $\boldsymbol{v} \in \mathcal{L}^2(\Omega, \text{co}\,(\Delta_\times))$ be arbitrary, and set $q_k^\lambda(x) := \hat{c}(x)/d$. Then

$$\sum_{k=1}^{d} \sum_{\lambda \in \Lambda_k} q_k^\lambda(x) v_k^\lambda(x) = \sum_{k=1}^{d} \frac{\hat{c}(x)}{d} \sum_{\lambda \in \Lambda_k} v_k^\lambda(x) = \sum_{k=1}^{d} \frac{\hat{c}(x)}{d} = \hat{c}(x), \tag{8.14}$$

and $\sum_k q_k^{\gamma_k}(x) = \hat{c}(x) \leq c^\gamma(x)$ for all $\gamma \in \Gamma$ and $x \in \Omega$, so $\boldsymbol{q} \in \mathcal{Q}$. This shows that $R_{\text{data}}(\boldsymbol{v}) \geq \int_\Omega \hat{c}(x)\,\mathrm{d}x$, which is the minimum of E_{data} for binary functions. ∎

8.5. Proof of proposition 4.4. By convex duality [30], the convex envelope of E_{data} is given by the Legendre-Fenchel bi-conjugate E_{data}^{**}. The first convex conjugate E_{data}^{*} of E_{data} is given as

$$E_{\text{data}}^{*}(\boldsymbol{q}) = \sup_{\boldsymbol{v}} \int_\Omega \sum_{k=1}^{d} \sum_{\lambda \in \Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x \;-\; E_{\text{data}}(v). \tag{8.15}$$

Since E_{data} is finite only for binary $\boldsymbol{v}$, i.e. if $v_k^\lambda(x) = \chi_{\lambda=\gamma_k(x)}$ for some $\gamma : \Omega \to \Gamma$, this reduces to

$$F(\boldsymbol{q}) := E_{\text{data}}^{*}(\boldsymbol{q}) = \sup_{\gamma:\Omega\to\Gamma} \int_\Omega \left(\sum_{k=1}^{d} q_k^{\gamma_k(x)}(x) - c^{\gamma(x)}(x) \right) \mathrm{d}x. \tag{8.16}$$

The bi-conjugate is then

$$E_{\text{data}}^{**}(\boldsymbol{v}) = \sup_{\boldsymbol{q}} \int_\Omega \sum_{k=1}^{d} \sum_{\lambda \in \Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x \;-\; F(q). \tag{8.17}$$

For $\boldsymbol{q} \in \mathcal{L}^2(\Omega, \mathbb{R}^{N_1+\ldots+N_d})$ and $a \in \mathcal{L}^2(\Omega, \mathcal{R})$ define $\boldsymbol{q}_a \in \mathcal{L}^2(\Omega, \mathbb{R}^{N_1+\ldots+N_d})$ by $(q_a)_k^\lambda(x) := q_k^\lambda(x) + a_k(x)$. Then obviously

$$F(\boldsymbol{q}_a) = F(\boldsymbol{q}) + \int_\Omega \sum_{k=1}^{d} a_k(x)\,dx \tag{8.18}$$

Inserting $\boldsymbol{q}_a$ for $\boldsymbol{q}$ in (8.17) we obtain

$$E_{\text{data}}^{**}(\boldsymbol{v}) = \sup_{\boldsymbol{q},\,a} \int_\Omega \sum_{k=1}^{d} \sum_{\lambda \in \Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x \;-\; F(q) + \int_\Omega \sum_{k=1}^{d} a_k(x) \left(\sum_{\lambda \in \Lambda_k} v_k^\lambda(x) - 1 \right) \mathrm{d}x \tag{8.19}$$

Holding a $\boldsymbol{q}$ fixed and taking the supremum over a we see that in order for $E_{\text{data}}^{**}(v)$ to be finite, we necessarily must have

$$\sum_{\lambda \in \Lambda_k} v_k^\lambda(x) = 1 \qquad \text{for all } 1 \leq k \leq d, \text{ for a.e. } x \in \Omega. \tag{8.20}$$

We assume these equalities from now on, i.e. that $\boldsymbol{v} \in \mathcal{S}$ with $\mathcal{S}$ as defined in the proposition.

By the same way as we arrived at (8.19) we see that given (8.20) the expression in (8.17), over which the supremum is taken, does not change if we replace $\boldsymbol{q}$ by $\boldsymbol{q}_a$ for some a. Also, (8.18) shows that for each $\boldsymbol{q}$ and any fixed $\alpha \in \mathbb{R}$ we can find an a with $F(\boldsymbol{q}_a) = \alpha$. This combined, we obtain

$$E^{**}_{\mathrm{data}}(\boldsymbol{v}) = \sup_{\boldsymbol{q}\colon F(\boldsymbol{q})=\alpha} \int_\Omega \sum_{k=1}^d \sum_{\lambda\in\Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x \;-\; \alpha. \tag{8.21}$$

Bringing α on the left hand side and taking the supremum over all $\alpha \le 0$ we get

$$E^{**}_{\mathrm{data}}(\boldsymbol{v}) = \sup_{\boldsymbol{q}\colon F(\boldsymbol{q})\le 0} \int_\Omega \sum_{k=1}^d \sum_{\lambda\in\Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x. \tag{8.22}$$

This is almost the expression (4.10) for $R_{\mathrm{data}}(\boldsymbol{v})$. We only need to replace the integral constraints on $\boldsymbol{q}$ in $F(\boldsymbol{q}) \le 0$, with F in (8.16), with *pointwise* constraints. For this, note that in (8.16) the supremum over γ may be safely put inside the integral over Ω since the label space Γ is finite (after the assumed discretization). Therefore, $F(\boldsymbol{q}) \le 0$ is equivalent to

$$\int_\Omega \sup_{\gamma\in\Gamma} \Big(\sum_{k=1}^d q_k^{\gamma_k}(x) - c^\gamma(x) \Big)\,\mathrm{d}x \;\le\; 0. \tag{8.23}$$

Denoting the integrand by $a(x)$, this becomes equivalent to

$$\exists\, a : \Omega \to \mathbb{R} : \quad \int_\Omega a(x)\,\mathrm{d}x \le 0, \quad \sum_{k=1}^d q_k^{\gamma_k}(x) - c^\gamma(x) \;\le a(x) \quad \forall \gamma \in \Gamma,\, x \in \Omega. \tag{8.24}$$

Given a $\boldsymbol{q}$ satisfying these constraints for some a, define $\widehat{\boldsymbol{q}}$ by $\widehat{q}_k^\lambda(x) := q_k^\lambda(x) - a(x)/d$. Then $\widehat{\boldsymbol{q}}$ satisfies (8.24) with $a \equiv 0$, i.e. $\widehat{\boldsymbol{q}} \in \mathcal{Q}$ with the constraint set $\mathcal{Q}$ in (4.11). Furthermore,

$$\begin{aligned}
\int_\Omega \sum_{k=1}^d \sum_{\lambda\in\Lambda_k} \widehat{q}_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x &= \int_\Omega \sum_{k=1}^d \sum_{\lambda\in\Lambda_k} \Big(q_k^\lambda(x) - \frac{a(x)}{d} \Big) v_k^\lambda(x)\,\mathrm{d}x \\
&= \int_\Omega \sum_{k=1}^d \sum_{\lambda\in\Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x \;-\; \int_\Omega a(x)\,\mathrm{d}x \\
&\ge \int_\Omega \sum_{k=1}^d \sum_{\lambda\in\Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x
\end{aligned} \tag{8.25}$$

using first (8.20) and then $\int_\Omega a(x) \le 0$. Thus, among all $\boldsymbol{q}$ with $F(\boldsymbol{q}) \le 0$ or, equivalenty, with (8.24) the expression in the supremum (8.22) will be largest if we choose $a \equiv 0$ in (8.24). Hence,

$$E^{**}_{\mathrm{data}}(\boldsymbol{v}) = \sup_{\boldsymbol{q}\in\mathcal{Q}} \int_\Omega \sum_{k=1}^d \sum_{\lambda\in\Lambda_k} q_k^\lambda(x) v_k^\lambda(x)\,\mathrm{d}x = R_{\mathrm{data}}(\boldsymbol{v}) \tag{8.26}$$

for $\boldsymbol{v} \in \mathcal{S}$. □

8.6. Proof of proposition 5.2. We can enforce a piecewise constant labeling u_k, if we enforce the approximate gradient ∇u_k to be constant zero. In (3.12), this can be achieved by setting $h_k(x, u_k(x), \nabla u_k(x)) = c\,|\nabla u_k|$ with a constant $c > 0$, and then letting $c \to \infty$ to enforce $\nabla u_k \equiv 0$ on $\Omega \setminus S_{u_k}$. Inserting the convex conjugate $h_k^*(x, \lambda, q) = \delta_{\{|q| \leq c\}}$, we find that the conditions in (4.4) now reduce to

$$b^\lambda \geq 0, \quad \left|\partial_\lambda \boldsymbol{p}^\lambda\right|_2 \leq c, \left|\boldsymbol{p}^\lambda - \boldsymbol{p}^\mu\right|_2 \leq d_k(\lambda, \mu). \tag{8.27}$$

The supremum over $b^\lambda \geq 0$ is easily eliminated from (4.3) since $v_k^\lambda \geq 0$, i.e. $-b^\lambda v_k^\lambda \leq 0$ with 0 being the maximum possible value. The second constraint in (8.27) follows from the third if we choose $c \geq \max_{\lambda > \mu} \frac{d_k(\lambda,\mu)}{|\lambda - \mu|}$. Thus we arrive at (5.7) with the set C_k as claimed in the proposition. ∎

8.7. Proof of proposition 5.3. The claim follows from our general formulation (5.7) with a special choice of the dual variables $\boldsymbol{p}$ together with additional relaxations of the equations in C_k. The special form for $\boldsymbol{p}^\lambda$ we choose is

$$\boldsymbol{p}^\lambda = \sum_{i=1}^{M_k} \boldsymbol{a}_{k,i}^\lambda \boldsymbol{q}_i, \tag{8.28}$$

with $\boldsymbol{q} : \Omega \times \{1, \dots, M_k\} \to \mathbb{R}^n$ such that $|\boldsymbol{q}|_2 \leq 1$ and the vectors $\boldsymbol{a}_k^\lambda \in \mathbb{R}^{M_k}$ which define the Euclidean representation of d_k, see equation (5.1). This is only a subset of possible $\boldsymbol{p} \in C_k$ in Proposition 5.3. The constraint on $\boldsymbol{p}$ in (5.7) is satisfied, since by the Cauchy-Schwarz inequality and the definition of the representation,

$$\begin{aligned} \left|\boldsymbol{p}^\lambda - \boldsymbol{p}^\mu\right|_2 &= \left| \sum_{i=1}^{M_k} (\boldsymbol{a}_{k,i}^\lambda - \boldsymbol{a}_{k,i}^\mu) \boldsymbol{q}_i \right|_2 \\ &\leq \sqrt{\sum_{i=1}^{M_k} (\boldsymbol{a}_{k,i}^\lambda - \boldsymbol{a}_{k,i}^\mu)^2} \cdot \sqrt{\sum_{i=1}^{M_k} |\boldsymbol{q}_i|_2^2} \\ &= \left|A_k \boldsymbol{e}^\lambda - A_k \boldsymbol{e}^\mu\right|_2 |\boldsymbol{q}|_2 \leq d_k(\lambda, \mu). \end{aligned} \tag{8.29}$$

Plugging (8.28) into (5.7) we obtain the desired result

$$\begin{aligned} J_k(\boldsymbol{v}_k) &\geq \sup_{|\boldsymbol{q}|_2 \leq 1} \left\{ \sum_{\lambda \in \Lambda_k} \int_\Omega \left(\sum_{i=1}^{M_k} \boldsymbol{a}_{k,i}^\lambda \boldsymbol{q}_i \right) \cdot \nabla v_k^\lambda \, \mathrm{d}x \right\} \\ &= \sup_{|\boldsymbol{q}|_2 \leq 1} \left\{ \int_\Omega \sum_{i=1}^{M_k} \boldsymbol{q}_i \cdot \nabla \left(\sum_{\lambda \in \Lambda_k} \boldsymbol{a}_{k,i}^\lambda v_k^\lambda \right) \mathrm{d}x \right\} \\ &= \sup_{|\boldsymbol{q}|_2 \leq 1} \left\{ \int_\Omega \sum_{i=1}^{M_k} \boldsymbol{q}_i \cdot \nabla (A_k \boldsymbol{v}_k)_i \, \mathrm{d}x \right\} \\ &= \mathrm{TV_v}(A_k \boldsymbol{v}_k). \end{aligned} \tag{8.30}$$

The inequality in the first step is a consequence of choosing the special form of $\boldsymbol{p}$'s, thus reducing the set over which the supremum is taken. ∎

8.8. Proof of proposition 6.1. Both J and R_{data} are support functionals of convex sets in the Hilbert space $\mathcal{L} := \mathcal{L}^2(\Omega, \mathbb{R}^{N_1+\ldots+N_d})$: equation (6.3) shows that the regularizer J is the support functional of $K(\mathcal{C})$, while we can see from definition (4.10) that the data term R_{data} is the support functional of $\mathcal{Q}$. It follows that both J and R_{data} are lower semicontinuous and convex on $\mathcal{L}$. The set $\mathcal{D}$ is closed, thus its indicator function $\delta_{\mathcal{D}}$ is also convex and closed, furthermore $\delta_{\mathcal{D}}$ is coercive since $\mathcal{D}$ is bounded. From the above, it follows that the functional

$$\boldsymbol{v} \mapsto J(\boldsymbol{v}) + R_{\text{data}}(\boldsymbol{v}) + \delta_{\mathcal{D}}(\boldsymbol{v}) \tag{8.31}$$

is closed and coercive. Since being closed is equivalent to being lower semicontinuous in the Hilbert space topology of $\mathcal{L}$, these properties imply the existence of a minimizer in $\mathcal{L}$, see theorems 3.2.5 and 3.3.3 in [3], which must necessarily lie in $\mathcal{D}$. Since neither functional is strictly convex, the solution is in general not unique. □