Convex Relaxation for Multilabel Problems with Product Label Spaces

Bastian Goldluecke and Daniel Cremers

Computer Vision Group, TU Munich

Abstract. Convex relaxations for continuous multilabel problems have attracted a lot of interest recently [1–5]. Unfortunately, in previous methods, the runtime and memory requirements scale linearly in the total number of labels, making them very inefficient and often unapplicable for problems with higher dimensional label spaces. In this paper, we propose a reduction technique for the case that the label space is a product space, and introduce proper regularizers. The resulting convex relaxation requires orders of magnitude less memory and computation time than previously, which enables us to apply it to large-scale problems like optic flow, stereo with occlusion detection, and segmentation into a very large number of regions. 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 of typically 5% of the global optimum.

1 Introduction

1.1 The Multi-labeling Problem

A multitude of computer vision problems like segmentation, stereo reconstruction and optical flow estimation can be formulated as multi-label problems. In this class of problems, we want to assign to each point x in an image domain $\Omega \subset \mathbb{R}^n$ a *label* from a discrete set $\Gamma = \{1, \dots, N\} \subset \mathbb{N}$. Assigning the label $\gamma \in \Gamma$ to x is associated with the *cost* $c_\gamma(x) \in \mathbb{R}$. In computer vision applications, the local costs usually denote how well a given labeling fits some observed data. They can be arbitrarily complex, for instance derived from statistical models or complicated local matching scores. We only assume that the cost functions c_γ lie in the Hilbert space of square integrable functions $\mathcal{L}^2(\Omega)$. Aside from the local costs, each possible labeling $g : \Omega \to \Gamma$ is penalized by a *regularization term* $J(g) \in \mathbb{R}$. The regularizer J represents our knowledge about which label configurations are a priori more likely. Frequently, it enforces some form of spacial coherence. In this paper, we are above all interested in regularizers which penalize proportionally to the length of the interface between regions with different labels γ, χ and a metric $d(\gamma, \chi)$ between the associated labels.

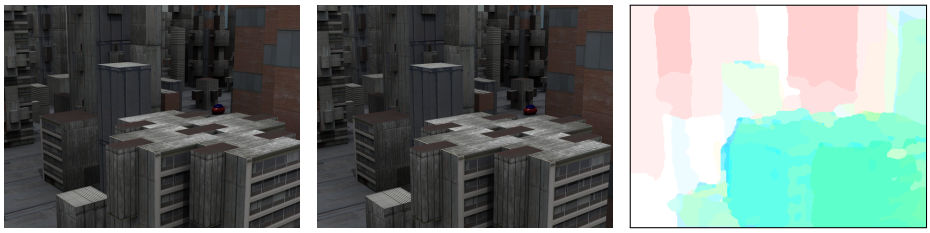


Fig. 1. *The proposed relaxation method can approximate the solution to multi-labeling problems with a huge number of possible labels by globally solving a convex relaxation model. This example shows two images and the optic flow field between the two, where flow vectors were assigned from a possible 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. See Fig. 7 for the color code of the flow vectors.*

The goal is to find a labeling $g : \Omega \rightarrow \Gamma$ which minimizes the sum of the total costs and the regularizer, i.e.

$$\underset{g \in \mathcal{L}^2(\Omega,\Gamma)}{\operatorname{argmin}} \; J(g) + \int_\Omega c_{g(x)}(x) \, \mathrm{d}x \, . \tag{1}$$

1.2 Discrete approaches

It is well known that in the fully discrete setting, the minimization problem (1) is equivalent to maximizing a Bayesian posterior probability, where the prior probability gives rise to the regularizer [6]. The problem can be stated in the framework of Markov Random Fields [7] and discretized using a graph representation, where the nodes denote discrete pixel locations and the edges encode the energy functional [8].

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) can be found by computing a minimum cut [9, 10]. For multi-label problems, one can approximate a solution for example by solving a sequence of binary problems (α-expansions) [11, 12], or linear programming relaxations [13]. Exact solutions to multi-label problems can only be found in some special cases, notably [14], where a cut in a multi-layered graph is computed in polynomial time to find a global optimum. The construction is restricted to convex iteraction terms with respect to a linearly ordered label set.

However, in many important scenarios the label space can not be ordered, or a non-convex regularizer is more desireable 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 [11]. Furthermore, it is known that the graph-based discretization induces an anisotropy, so that the

solutions suffer from metrication errors [15]. It is therefore interesting to investigate continuous approaches as a possible alternative.

1.3 Continuous approaches

Continuous approaches deal with the multi-label problem by transforming it into a continuous convex problem, obtaining the globally optimal solution, and projecting the continuous solution back onto the original discrete space of labels. Depending on the class of problem, it can be possible to obtain globally optimal solutions to the original discrete minimization problem.

As in the discrete setting, it is possible to solve the two-label problem in a globally optimal way by minimizing a continuous convex energy and subsequent thresholding [2]. In the case of convex interaction terms and a linearly ordered set of labels, there also exists a continuous version of [14] to obtain globally optimal solutions [3]. For the general multi-label case, however, there is no relaxation known which leads to globally optimal solutions of the discrete problem. Currently the most tight relaxation is [4]. The theoretical basis of the reduction technique introduced in this paper is the slightly more transparent formulation introduced in [5] and further generalized in [1], but it can be easily adapted to the framework [4] as well.

The convex relaxation described in [1, 5] works as follows. Instead of looking for g 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 $g(x) = \gamma$. To make sure that a unique label is assigned to each point, only one of the indicator functions can have the value one. Thus, we restrict optimization to the space

$$\mathcal{U}_\Gamma := \left\{ (u_\gamma)_{\gamma\in\Gamma} \;:\; u_\gamma \in \mathcal{L}^2(\Omega,\{0,1\}) \text{ and } \sum_{\gamma\in\Gamma} u_\gamma(x) = 1 \text{ for all } x \in \Omega \right\} . \tag{2}$$

Let $\langle\cdot,\cdot\rangle$ denote the inner product on the Hilbert space $\mathcal{L}^2(\Omega)$, then problem (1) can be written in the equivalent form

$$\operatorname*{argmin}_{\boldsymbol{u}\in\mathcal{U}_\Gamma} J(\boldsymbol{u}) + \sum_{\gamma\in\Gamma} \langle u_\gamma, c_\gamma\rangle \ , \tag{3}$$

where we use bold face notation $\boldsymbol{u}$ for vectors $(u_\gamma)_{\gamma\in\Gamma}$ indexed by elements in Γ. We use the same symbol J to also denote the regularizer on the reduced space. Its definition requires careful consideration, see Section 3.

1.4 Contribution: Product label spaces

In this work, we discuss label spaces which can be written as a product of a finite number d of discrete spaces, $\Gamma = \Lambda_1 \times \cdots \times \Lambda_d$. Let N_j be the number of elements in Λ_j, then the total number of labels is $N = N_1 \cdot \ldots \cdot N_d$. In the formulation (3), we optimize over a number of N binary functions, which can

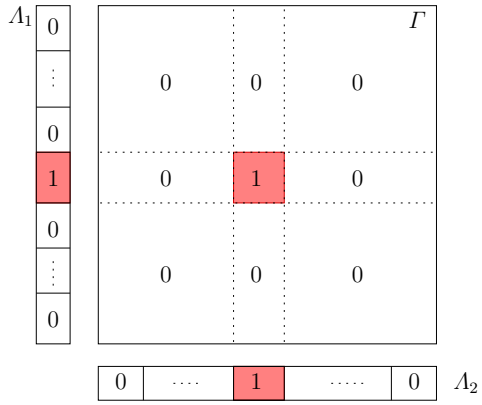


Fig. 2. *The central idea of the reduction technique is that if a single indicator function in the product space Γ takes the value 1, then this is equivalent to setting an indicator function in each of the factors Λ_j. The memory reduction stems from the fact that there are much more labels in Γ than in all the factors Λ_j combined.*

be rather large in practical problems. In order to make problems of this form feasible to solve, we present a further reduction which only requires $N_1 + \cdots + N_d$ binary functions - a linear instead of an exponential growth.

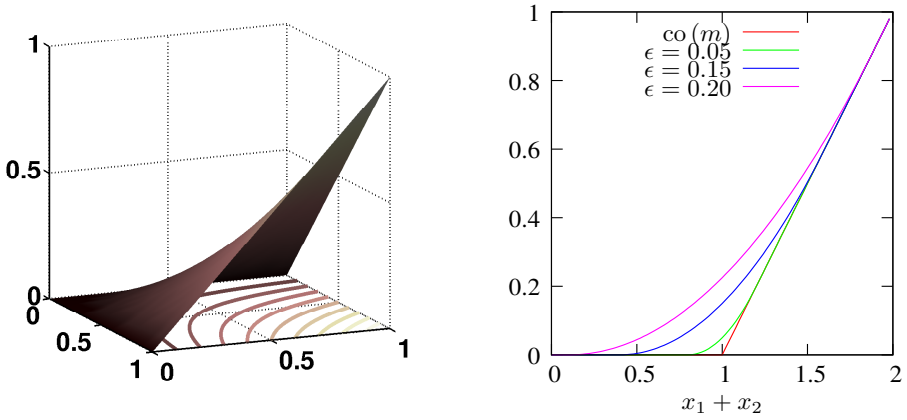
We will show that with our novel reduction technique, it is possible to efficiently solve convex relaxations to multi-label problems which are far too large to approach with previously existing techniques. A prototypical example is optic flow, where a typical total number of labels is around 32^2 for practical problems, for which we only require 64 indicator functions instead of 1024. However, the proposed method applies to a much larger class of labelling problems. A consequence of the reduction in variable size is a disproportionately large cut in required runtime, which also makes our method much faster.

2 Relaxations for Product Label Spaces

2.1 Product Label Spaces

As previously announced, from now on we assume that the space of labels is a product of a finite number d of discrete spaces, $\Gamma = \Lambda_1 \times \cdots \times \Lambda_d$, with $|\Lambda_j| = N_j$. To each label $\lambda \in \Lambda_j, 1 \leq j \leq d$, we associate an indicator function u^j_λ. Thus, optimization will take place over the reduced space of functions

$$\mathcal{U}^\times_\Gamma := \Big\{ (u^j_\lambda)_{1 \leq j \leq d, \lambda \in \Lambda_j} \; : u^j_\lambda \in \mathcal{L}^2(\Omega, \{0,1\}) \text{ and } \sum_{\lambda \in \Lambda_j} u^j_\lambda(x) = 1 \text{ for all } x \in \Omega, 1 \leq j \leq d \Big\}. \tag{4}$$



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

(b) Convex envelope $\mathrm{co}\,(m)$ and mollified versions for different ϵ

Fig. 3. *Product function and its mollified convex envelope for the case* $d = 2$.

We use the short notation $\boldsymbol{u}^\times$ for a tuple $(u^j_\lambda)_{1\le j\le d, \lambda\in\Lambda_j}$. Note that such a tuple consists indeed of exactly $N_1 + ... + N_d$ binary functions. The following proposition illuminates the relationship between the function spaces $\mathcal{U}_\Gamma$ and $\mathcal{U}^\times_\Gamma$.

Proposition 1. *A bijection* $\boldsymbol{u}^\times \mapsto \boldsymbol{u}$ *from* $\mathcal{U}^\times_\Gamma$ *onto* $\mathcal{U}_\Gamma$ *is defined by setting*

$$u_\gamma := u^1_{\gamma_1} \cdot ... \cdot u^d_{\gamma_d}\,,$$

for all $\gamma = (\gamma_1, ..., \gamma_d) \in \Gamma$.

This is easy to see visually, Figure 2.1. A formal proof can be found in the appendix. With this new function space, another equivalent formulation to (1) and (3) is

$$\operatorname*{argmin}_{\boldsymbol{u}^\times\in\mathcal{U}^\times_\Gamma} J(\boldsymbol{u}^\times) + \sum_{\gamma\in\Gamma} \left\langle u^1_{\gamma_1} \cdot ... \cdot u^d_{\gamma_d}, c_\gamma \right\rangle\,. \tag{5}$$

Note that 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 again convex.

2.2 Convex Relaxation

Two steps have to be taken to relax (5) to a convex problem. In a first step, we replace the multiplication function $m(u^1_{\gamma_1}, ..., u^d_{\gamma_d}) := u^1_{\gamma_1} \cdot ... \cdot u^d_{\gamma_d}$ with a convex function. In order to obtain a tight relaxation, we first move to the convex envelope $\mathrm{co}\,(m)$ of m. Analyzing the epigraph of m, Fig. 3(a) shows that

$$\mathrm{co}\,(m)\,(x_1, ..., x_d) = \begin{cases} 1 & \text{if } x_1 = ... = x_d = 1, \\ 0 & \text{if any } x_j = 0. \end{cases} \tag{6}$$

This means that if in the functional, m is replaced by the convex function $\operatorname{co}(m)$, 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.

In order to be able to solve the new problem in practice, we replace $\operatorname{co}(m)$ 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

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

This is a piecewise linear function of the sum of the arguments, i.e symmetric in x_1 and x_2, see Fig. 3(b). We smoothen the kink by replacing $\operatorname{co}(m)$ with

$$\operatorname{co}(m)_\epsilon(x_1, x_2) = \begin{cases} 0 & \text{if } x_1 + x_2 \leq 1 - 4\epsilon \\ \frac{1}{16\epsilon}(x_1 + x_2 - (1 - 4\epsilon))^2 & \text{if } 1 - 4\epsilon < x_1 + x_2 < 1 + 4\epsilon \\ 1 & \text{if } x_1 + x_2 \geq 1 + 4\epsilon \end{cases}$$

This function does not satisfy the above condition (6) exactly, but only fulfills the less tight

$$\operatorname{co}(m)_\epsilon(x_1, \ldots, x_d) \begin{cases} = 1 & \text{if } x_1 = \cdots = x_d = 1, \\ \leq \epsilon & \text{if any } x_j = 0. \end{cases} \tag{7}$$

The following Theorem shows that the solutions of the smoothened energy converge to the solutions of the original energy as $\epsilon \to 0$. After discretization, this means that we obtain an exact solution to the binary problem if we choose ϵ small enough, since the problem is combinatorial and the number of possible configurations finite.

Theorem 1. *Let $\epsilon > 0$ and $\operatorname{co}(m)_\epsilon$ satisfy condition (7). Let $\boldsymbol{u}_0^\times$ be a solution to problem (5), and*

$$\boldsymbol{u}_\epsilon^\times \in \operatorname*{argmin}_{\boldsymbol{u}^\times \in \mathcal{U}_\Gamma^\times} J(\boldsymbol{u}^\times) + \sum_{\gamma \in \Gamma} \left\langle \operatorname{co}(m)_\epsilon(u^1_{\gamma_1}, ..., u^d_{\gamma_d}), c_\gamma \right\rangle . \tag{8}$$

Then

$$\left| E_\epsilon(\boldsymbol{u}_\epsilon^\times) - E(\boldsymbol{u}_0^\times) \right| \leq |\Omega| \sum_{\gamma \in \Gamma} \|c_\gamma\|_\infty \, \epsilon \,, \tag{9}$$

where E and E_ϵ are the energies of the original problem (5) and smoothened problem (8), respectively.

The proof can be found in the appendix. The key difference of (8) compared to (5) is that the data term is now a convex function.

In the second step of the convex relaxation, we have to make sure the domain of the optimization is a convex set. Thus $\mathcal{U}_\Gamma^\times$ is replaced by its convex

hull co $\left(\mathcal{U}_\Gamma^\times\right)$. This just means that the domain of the functions (u_λ^j) is extended to the continuous interval $[0,1]$. The final relaxed problem which we are going to solve is now to find

$$\underset{\boldsymbol{u}^\times \in \mathrm{co}\left(\mathcal{U}_\Gamma^\times\right)}{\operatorname{argmin}} \; J(\boldsymbol{u}^\times) + \sum_{\gamma \in \Gamma} \left\langle \mathrm{co}\,(m)_\epsilon\,(u_{\gamma_1}^1, ..., u_{\gamma_d}^d), c_\gamma \right\rangle \,. \tag{10}$$

2.3 Numerical Method

With a suitable choice of convex regularizer J, problem (10) is a continuous convex problem with a convex and differentiable data term. In other relaxation methods, one usually employs fast primal-dual schemes [16, 17] to solve the continuous problem. However, those are only applicable to linear data terms. Fortunately, the derivative of the data term is Lipschitz-continuous with Lipschitz constant $L = \frac{1}{8\epsilon} \sum_\gamma \|c_\gamma\|_2$. If we take care to choose a lower semi-continuous J, we are thus in a position to apply the FISTA scheme [18] to the minimization of (10). It is much faster than for example direct gradient descent, with a provable quadratic convergence rate. The remaining problems are how to choose a correct regularizer, and how to get back from a possibly non-binary solution of the relaxed problem to a solution of the original problem.

2.4 Obtaining a solution to the original problem

Let $\hat{\boldsymbol{u}}^\times$ be a solution to the relaxed problem (10). Thus, the functions $\hat{u}_\lambda^j$ might have values in between 0 and 1. In order to obtain a feasible solution to the original problem (1), we just project back to the space of allowed functions. The function $\hat{g} \in \mathcal{L}^2(\Omega, \Gamma)$ closest to $\hat{\boldsymbol{u}}^\times$ is given by setting

$$\hat{g}(x) = \underset{\gamma \in \Gamma}{\operatorname{argmax}}\, \hat{u}_{\gamma_1}^1(x) \cdot ... \cdot \hat{u}_{\gamma_d}^d(x)\,,$$

i.e. we choose the label where the combined indicator functions have the highest value.

We cannot guarantee that the solution $\hat{g}$ is indeed a global optimum of the original problem (1), since there is nothing equivalent to the thresholding theorem [2] 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) must lie somewhere between the energies of $\hat{\boldsymbol{u}}^\times$ and $\hat{g}$.

3 Regularization

The following construction of a family of regularizers is analogous to [1], but extended to accomodate product label spaces. An element $\boldsymbol{u}^\times \in \mathcal{U}_\Gamma^\times$ can be viewed as a map in $\mathcal{L}^2(\Omega, \Delta^\times)$, where

$$\Delta^\times = \Delta^1 \times ... \times \Delta^d \subset \mathbb{R}^{N_1 + ... + N_d}$$

and

$$\Delta^i = \left\{ \boldsymbol{x} \in \{0,1\}^{N_i} \;:\; \sum_{j=1}^{N_i} x_j = 1 \right\}.$$

is the set of corners of the standard $(k-1)$-simplex. As shown previously, there is a one-to-one correspondence between elements in $\Delta^\times$ and the labels in Γ.

We will now construct a familiy of regularizers $J : \operatorname{co}\left(\mathcal{U}_\Gamma^\times\right) \to \mathbb{R}$ and afterwards demonstrate that it is well suited to the problem at hand. For this, we impose a metric d on the space Γ of labels. A current limitation is that we can only handle the case of separable metrics, i.e. d must be of the form

$$d(\gamma, \chi) = \sum_{i=1}^{d} d_i(\gamma_i, \chi_i), \tag{11}$$

where each d_i is a metric on Δ^i. We further assume that each d_i has an Euclidean representation. This means that each label $\lambda \in \Delta^i$ shall be *represented* by an r_i-dimensional vector $a_\lambda^i \in \mathbb{R}^{r_i}$, and the distance d_i defined as Euclidean distance between the representations,

$$d(\lambda, \mu) = |a_\lambda - a_\mu|_2 \text{ for all } \lambda, \mu \in \Delta^i. \tag{12}$$

The goal in the construction of J is that the higher the distance between labels, the higher shall be the penalty imposed by J. To make this idea precise, we introduce the linear mappings $A_i : \operatorname{co}\left(\Delta^i\right) \to \mathbb{R}^{r_i}$ which map labels onto their representations,

$$A_i(\lambda) = a_\lambda^i \text{ for all } \lambda \in \Delta^i.$$

When the labels are enumerated, then in matrix notation, the vectors a_γ^i become exactly the columns of A_i, which shows the existence of this map.

We can now define the regularizer as

$$J(\boldsymbol{u}^\times) := \sum_{i=1}^{d} \mathrm{TV}_\mathrm{v}^i(A_i \boldsymbol{u}^i), \tag{13}$$

where TV_v^i is the vectorial total variation on $\mathcal{L}^2(\Omega, \mathbb{R}^{r_i})$. The following theorem shows why the above definition makes sense.

Theorem 2. *The regularizer J defined in* (13) *has the following properties:*

1. *J is convex and positively homogenous on $\operatorname{co}\left(\mathcal{U}_\Gamma^\times\right)$.*
2. *$J(\boldsymbol{u}^\times) = 0$ for any constant labeling $\boldsymbol{u}^\times$.*
3. *If $S \subset \Omega$ has finite perimeter $\operatorname{Per}(S)$, then for all labels $\gamma, \chi \in \Gamma$,*

$$J(\gamma 1_S + \chi 1_{S^c}) = d(\gamma, \chi) \operatorname{Per}(S),$$

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

# of Pixels $P = P_x \times P_y$	# Labels $N_1 \times N_2$	Memory [Mb] Previous	Memory [Mb] Proposed	Runtime [s] Previous	Runtime [s] Proposed
320×240	8×8	112	28	196	6
320×240	16×16	450	56	*	21
320×240	32×32	1800	112	*	75
320×240	64×64	7200	225	-	314
640×480	8×8	448	112	789	22
640×480	16×16	1800	224	*	80
640×480	32×32	7200	448	-	297
640×480	64×64	28800	900	-	1112

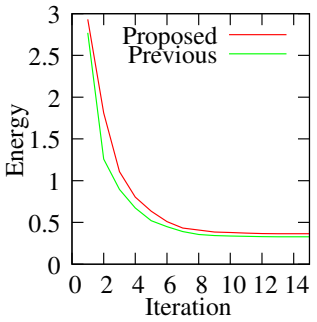


Fig. 4. *The table shows the total amount of memory required for a FISTA implementation of the previous and proposed methods depending on the size of the problem. Also shown is the total runtime for 15 iterations, which usually suffices for convergence. Numbers shown in red cannot be stored within even the largest of todays CUDA capable cards, so an efficient parallel implementation is not possible. Failures marked with a "∗" are due to another limitation: the shared memory is only sufficient to store the temporary variables for the simplex projection up until dimension* 128*. In the graph, we see a comparison of the convergence rate between the original scheme and the proposed scheme. Despite requiring significantly less memory and runtime, the relaxation is still sufficiently tight to arrive at an almost similar solution.*

The theorem is proved in the appendix. More general classes of metrics on the labels can also be used, see [1]. For the sake of simplicity, we only included the most important example of distances with Euclidean representations. This class includes, but is not limited to, the following special cases:

- The Potts or uniform distance, where $d_i(\lambda, \mu) = 1$ if and only if $\lambda = \mu$, and zero otherwise. This distance function can be achieved by setting $a^i_\lambda = \frac{1}{2} e_\lambda$, where $(e_\lambda)_{\lambda \in \Lambda_i}$ is an orthonormal basis in $\mathbb{R}^{N_i}$. All changes between labels are penalized equally.
- The typical case is that the a^i_λ 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 $\mathbb{R}^2$. The representations a^1_λ, a^2_μ are just real numbers, denoting the possible components of the flow vectors in x and y-direction, respectively. The Euclidean distance is a natural distance on the components to regularize the flow field, corresponding to the regularizer of the TV-L^1 functional in [19].

The convex functional we wish to minimize is now fully defined, including the regularizer. The ROF type problems with the vectorial total variation as a regularizer, which are at the core of the resulting FISTA scheme, can be minimized with algorithms in [20]. For the also required backprojections onto simplices we recommend the method in [21]. Thus, we can turn our attention towards computing a minimizer in practice. In the remaining section, we will apply the framework to a variety of computer vision problems.

Fig. 5. *Results for the multi-label segmentation. The input image on the left was labelled with* 8×8 *labels in the hue and lightness components of HSL color space. The label distance is set so that smoothing is stronger in darker regions, which creates an interesting visual effect.*

4 Experiments

We implemented the proposed algorithm for the case $d = 2$ on parallel processing GPU architecture using the CUDA programming language, and performed a variety of experiments, with completely different data terms and regularizers. All experiments were performed on an nVidia Tesla C1060 card with 4GB of memory.

When the domain Ω is discretized into P pixels, the primal and dual variables required for the FISTA minimization scheme are represented as matrices. In total, we have to store $P \cdot (N_1 + ... + N_d)$ floating point numbers for the primal variables, and $Pn \cdot (r_1 + ... + r_d)$ floating point numbers for the dual variables. In contrast, without using our reduction scheme, this number would be as high as $P \cdot N_1 \cdot ... \cdot N_d$ for the primal variables and $Pn \cdot r_1 \cdot ... \cdot r_d$ for the dual variables, respectively. For the FISTA scheme, we need space for four times the primal variables in total, so we end up with the total values shown in Fig. 4. Thus, problems with large number of labels can only be handled with the proposed reduction technique.

4.1 Multi-label Segmentation

For the first example, we chose one with a small label space, so that we can compare the convergence rate and solution energy of the previous method [1] with the proposed one. We perform a segmentation of an image based on the HSL color space. The hue and lightness values of the labeling are taken from the discrete sets of equidistant labels Λ_1 and Λ_2, respectively. Their size is $|\Lambda_1| = |\Lambda_2| = 8$, so there are 64 labels in total, which can still be handled by the old method as well, albeit barely. The labels shall be as close as possible to the original image values, so the cost function penalizes the $\mathcal{L}^1$-distance in HSV color space. We choose the regularizer so that the penalty for discontinuities is proportionally larger in regions with higher lightness. The relaxation constant ϵ is reduced from 0.2 to 0.05 during the course of the iterations. The result can be

Fig. 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: (a) Left input image I_L. (b) Right input image I_R. (c) Computed disparity and occlusion map, red areas denote occluded pixels.*

seen in Fig. 5, while a comparison of the respective convergence rates are shown in the graph in Fig. 4. The proposed method, despite requiring only a fraction of the memory and computation time, achieves a visually similar result with only a slightly higher energy. Note that the runtime of our method is far lower, since the simplex projection becomes disproportionally more expensive if the length of the vector is increased.

4.2 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 wether 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 Λ_2 for the occlusion map. We use the technique in [1] to approximate a truncated linear smoothness penalty on the disparity values. A Potts regularizer is imposed for the occlusion map. The distance on the label space thus becomes

$$d(\gamma, \chi) = s_1 \min(t_1, |\gamma_1 - \chi_1|) + s_2 \, |\gamma_2 - \chi_2| \ , \tag{14}$$

with suitable weights $s_1, s_2 > 0$ and threshold $t_1 > 0$. 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

$$c_\gamma(x, y) = \begin{cases} c_{occ} & \text{if } \gamma_2 = 1, \\ \|I_L(x, y) - I_R(x - \lambda_1, y)\|_2 & \text{otherwise.} \end{cases} \tag{15}$$

The result for the "Moebius" test pair from the Middlebury benchmark is shown in Fig. 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 597 seconds.

First image I_0 Second image I_1 Flow field and color code

Fig. 7. *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.*

4.3 Optic Flow

In the final test, 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 a truncated linear penalty on the component distance, i.e.

$$d(\gamma, \chi) = s \min(t, |\gamma_1 - \chi_1|) + s \min(t, |\gamma_2 - \chi_2|)\,, \tag{16}$$

with a suitable weight $s > 0$ and threshold $t > 0$. The cost function just compares pointwise pixel colors in the images, i.e.

$$c_\gamma(x, y) = \|I_0(x, y) - I_1(x + \gamma_1, y + \gamma_2)\|_2\ . \tag{17}$$

Results can be observed in Fig. 1 and 7. Due to the global optimization of a convex energy, we can successfully capture large displacements without having to implement a coarse-to-fine scheme. The number of labels is 50×50 at an image resolution of 640×480, so the memory requirements are so high that this problem is currently impossible to solve with previous convex relaxation techniques by a large margin, see Fig. 4. Total computation time using our method was 678 seconds. A comparison of the energies of the continuous and discretized solution shows that we are within 5% of the global optimum for all examples.

5 Conclusion

We have introduced a continuous convex relaxation for multi-label problems where the label space is a product space. 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 distance on the label space, a globally optimal solution of the relaxed problem and an efficient parallel GPU implementation with guaranteed convergence.

Because of the reduced memory requirements, we can successfully handle specific problems with very large number of labels, which could not be attempted with previous convex relaxation techniques. Among other examples we presented a convex relaxation for the optic flow functional with truncated linear penalizer on the distance between the flow vectors. To our knowledge, this is the first relaxation for this functional which can be optimized globally and efficiently.

References

1. Lellmann, J., , Becker, F., Schnörr, C.: 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)
2. Nikolova, M., Esedoglu, S., T.Chan: Algorithms for finding global minimizers of image segmentation and denoising models. SIAM Journal of Applied Mathematics **66** (2006) 1632–1648
3. Pock, T., Schoenemann, T., Graber, G., Bischof, H., Cremers, D.: A convex formulation of continuous multi-label problems. In: European Conference on Computer Vision (ECCV). (2008) 792–805
4. Pock, T., Chambolle, A., Bischof, H., Cremers, D.: A convex relaxation approach for computing minimal partitions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2009) 810–817
5. Zach, C., Gallup, D., Frahm, J., Niethammer, M.: Fast global labeling for real-time stereo using multiple plane sweeps. In: Vision, Modeling and Visualization. (2009) 243–252
6. Szeliski, R.: Bayesian modeling of uncertainty in low-level vision. International Journal of Computer Vision **5** (1990) 271–301
7. Kindermann, R., Snell, J.: Markov Random Fields and Their Applications. American Mathematical Society (1980)
8. Boykov, Y., Kolmogorov, V.: Computing geodesics and minimal surfaces via graph cuts. In: IEEE International Conference on Computer Vision (ICCV). (2003) 26–33
9. Greig, D., Porteous, B., Seheult, A.: Exact maximum a posteriori estimation for binary images. J. Royal Statistics Soc. **51** (1989) 271–279
10. Kolmogorov, V., Zabih, R.: What energy functions can be minimized via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **26** (2004) 147–159
11. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23** (2001) 1222–1239
12. Schlesinger, D., Flach, B.: Transforming an arbitrary min-sum problem into a binary one. Technical report, Dresden University of Technology (2006)
13. Wainwright, M., Jaakkola, T., Willsky, A.: Map estimation via agreement on trees: message-passing and linear programming. IEEE Trans. Inf. Theory **51** (2005) 3697–3717
14. Ishikawa, H.: Exact optimization for markov random fields with convex priors. IEEE Trans. Pattern Anal. Mach. Intell. **25** (2003) 1333–1336
15. Klodt, M., Schoenemann, T., Kolev, K., Schikora, M., Cremers, D.: An experimental comparison of discrete and continuous shape optimization methods. In: European Conference on Computer Vision (ECCV). (2008) 332–345
16. Popov, L.: A modification of the arrow-hurwicz method for search of saddle points. Math. Notes **28** (1980) 845–848

17. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Prog. **103** (2004) 127–152
18. Beck, A., Teboulle, M.: Fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sciences **2** (2009) 183–202
19. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime $TV - L^1$ optical flow. In: Pattern Recognition (Proc. DAGM). (2007) 214–223
20. Duval, V., Aujol, J.F., Vese, L.: Projected gradient based color image decomposition. In: Scale Space and Variational Methods in Computer Vision. (2009) 295–306
21. Michelot, C.: 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) 195–200

Appendix

Proof of Proposition 1. In order to proof the proposition, we have to show that the mapping induces a point-wise bijection from $\Delta^\times$ onto

$$\Delta = \left\{ \boldsymbol{x} \in \{0,1\}^N \;:\; \sum_{j=1}^{N} x_j = 1 \right\}.$$

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 $u^i_\lambda(x) = 1$ if $\lambda = \gamma_i$, and $u^i_\lambda(x) = 0$ otherwise. Then $\boldsymbol{u}(x) = \boldsymbol{u}^1(x) \cdot \ldots \cdot \boldsymbol{u}^d(x)$, as desired. To see that the map is one-to-one, we just count the elements in $\Delta^\times$. Since Δ^i contains N_i elements, the number of elements in $\Delta^\times$ is $N_1 \cdot \ldots \cdot N_d = N$, the same as in Δ. □

Proof of Theorem 1.

The regularizers of the original and smoothened problems are the same, so because of condition (7),

$$\left| E_\epsilon(\boldsymbol{u}^\times_\epsilon) - E(\boldsymbol{u}^\times_0) \right| \leq \left| \sum_{\gamma \in \Gamma} \int_\Omega \epsilon c_\gamma \, \mathrm{d}x \right| \leq |\Omega| \sum_{\gamma \in \Gamma} \|c_\gamma\|_\infty \, \epsilon \,. \tag{18}$$

This completes the proof. □

Proof of Theorem 2.

The first two claims are basic properties of the total variation. For the last claim, we combine Corollary 1 in [1] with the definition of the metric in equations (11) and (12) to find

$$\begin{aligned} J(\gamma 1_S + \chi 1_{S^c}) &= \sum_{i=1}^{d} \mathrm{TV}^i_{\mathrm{v}}(A_i(\gamma 1_S + \chi 1_{S^c})) = \sum_{i=1}^{d} \left| a^i_\gamma - a^i_\chi \right|_2 \mathrm{Per}(S) \\ &= d(\gamma, \chi) \, \mathrm{Per}(S). \end{aligned} \tag{19}$$

This completes the proof. □