# Generalized Ordering Constraints for Multilabel Optimization

Evgeny Strekalovskiy and Daniel Cremers
TU Munich, Germany

## Abstract

*We propose a novel framework for imposing label ordering constraints in multilabel optimization. In particular, label jumps can be penalized differently depending on the jump direction. In contrast to the recently proposed MRF-based approaches, the proposed method arises from the viewpoint of spatially continuous optimization. It unifies and generalizes previous approaches to label ordering constraints: Firstly, it provides a common solution to three different problems which are otherwise solved by three separate approaches [4, 10, 14]. We provide an exact characterization of the penalization functions expressible with our approach. Secondly, we show that it naturally extends to three and higher dimensions of the image domain. Thirdly, it allows novel applications, such as the* convex shape prior. *Despite this generality, our model is easily adjustable to various label layouts and is also easy to implement. On a number of experiments we show that it works quite well, producing solutions comparable and superior to those obtained with previous approaches.*

## 1. Introduction

### 1.1. Multilabeling and Ordering Constraints

Multilabel optimization is an important challenge in computer vision. It spans a great variety of problems, such as segmentation, stereo, optical flow and denoising. Among the first computational paradigms for efficiently solving multilabel problems was the graph cut approach of Ishikawa [6] for convex regularizers. For more general cost functions, Boykov et al. [2] introduced the concept of $\alpha$-expansion to approximate the hard multilabel problems through a sequence of binary problems. An approach based on primal dual linear programming was subsequently introduced by Komodakis and Tziritas [7]. Spatially continuous multilabel approaches were introduced in [1, 3, 8, 15].

A substantial generalization of penalty functions came about with the introduction of ordering constraints into the multilabel optimization. Penalizing label jumps differently depending on the jump direction allows to model specific label *layouts*. Liu et al. [10] showed how certain multilabel problems with ordering constraints could be solved using graph cuts. The five regions layout to segment indoor and outdoor images was introduced. Felzenszwalb and Veksler [4] introduced the tiered layout – a generalization of the five regions layout – and showed that it was solvable by dynamic programming. An entirely separate ordering constraint was introduced by the star shape prior of Veksler [14].

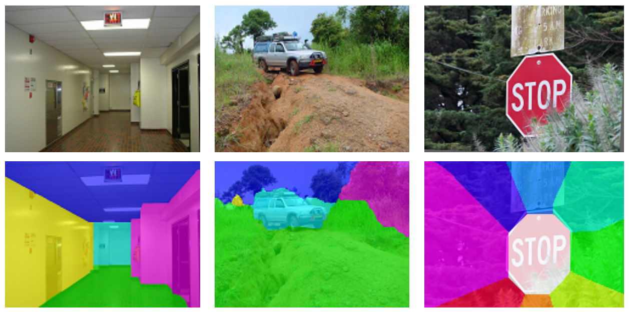
Figure 1. We propose a spatially continuous framework for label order constraints which unifies existing approaches such as the five regions layout (left) and the tiered layout (middle). It generalizes to novel applications such as a convex shape prior (right).

### 1.2. Contribution

We propose a novel general framework to incorporate ordering constraints. In contrast to the discrete graph cut or dynamic programming approaches of Liu, Veksler, Felzenszwalb and coworkers, the proposed approach comes from a totally different viewpoint of continuous optimization. We provide an exact characterization of the penalty functions expressible with our approach. In particular, the proposed method exhibits several favorable properties:

- We show in Sec. 4 that the three mentioned layout approaches are special cases of the proposed framework.
- We show that this framework allows applications beyond the above approaches, including tiered layout with four and more tiers, tiered layout with independent floating occlusions and shape priors for arbitrary *convex* shapes – see Figure 2.
- In contrast to existing approaches to label ordering constraints the proposed framework naturally extends to three and higher dimensions of the image domain.

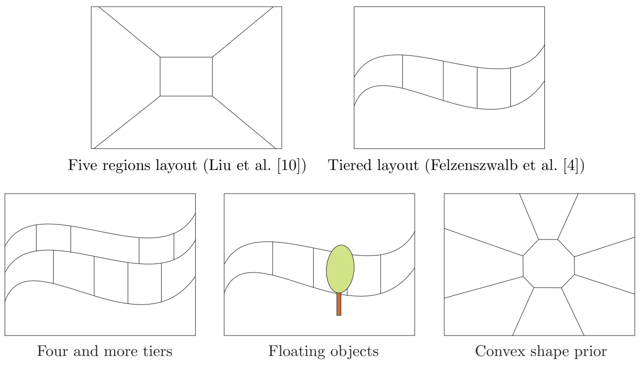
Five regions layout (Liu et al. [10])
Tiered layout (Felzenszwalb et al. [4])
Four and more tiers
Floating objects
Convex shape prior

Figure 2. Schematic summary of our main contributions.

- Despite its generality, the model is easily adjusted to various label layouts, it is easily implemented and provides results which are comparable and often superior to those of existing approaches.

## 2. General Framework

The general multilabel problem in image domain $\Omega \subset \mathbb{R}^m$, $m \geq 1$, with $n \geq 1$ labels consists in finding $n$ label indicator functions $u_1, \dots, u_n : \Omega \to \{0,1\}$ minimizing

$$\inf_u \left\{ \sum_{i=1}^{n} \int_\Omega \varrho_i(x) u_i(x)\, dx \;+\; R(u) \right\} \tag{2.1}$$

under the label-uniqueness constraint $\sum_{i=1}^n u_i(x) = 1$. Here, $\varrho_i(x)$ is the data term, i.e. the local cost of assigning label $i$ at image point $x \in \Omega$, and $R(u)$ a multilabel *regularizer* ensuring a spatial consistency of labels. To obtain a convex optimization problem, we relax the binary constraint $u_i(x) \in \{0,1\}$ to $u_i(x) \in [0,1]$ for all labels $i$.

### 2.1. The Novel Regularizer

We propose the following regularizer:

$$R(u) = \sup_{p \in C} \sum_{i=1}^{n} \int_\Omega \langle p_i(x), \nabla u_i(x) \rangle \, dx \tag{2.2}$$

with the convex set

$$C = \left\{ (p_i)_{i=1..n} : \Omega \to (\mathbb{R}^m)^n \,\middle|\, \langle p_j - p_i, \nu \rangle \leq d(i,j,\nu) \right\}. \tag{2.3}$$

The constraints in $C$ are taken pointwise for each $x \in \Omega$. The *label distance function* $d : \{1, \dots, n\}^2 \times \mathcal{S}^{m-1} \to \mathbb{R} \cup \{\infty\}$ gives the penalization if the multilabel assignment $u$ changes from label $i$ to label $j$ *in direction* $\nu$. The distance $d$ may also depend on the position $x \in \Omega$, thus enabling different regularizer weights at different image locations. $\langle .\,, . \rangle$ denotes the standard scalar product on $\mathbb{R}^m$ and $\mathcal{S}^{m-1} = \{ z \in \mathbb{R}^m \,|\, |z| = 1 \}$ the $(m-1)$-sphere.

There is a simple intuition behind the definition of the regularizer (2.2) with the constraint set (2.3). Suppose that at some point $x_0$ the labeling changes from label $i$ to label $j$ in direction $\nu$, i.e. locally we have $u_i(x) = 1$ for $\langle x - x_0, \nu \rangle \leq 0$ an $u_j(x) = 1$ otherwise. Passing through $x_0$, $u_i$ decreases from 1 to 0 and $u_j$ increases from 0 to 1. Thus, $\nabla u_i = -\nu$ and $\nabla u_j = \nu$ up to the delta function factor $\delta(\langle x - x_0, \nu \rangle)$. The integral (2.2) is thus locally $\langle p_i, \nabla u_i \rangle + \langle p_j, \nabla u_j \rangle = \langle p_j - p_i, \nu \rangle$ times the local boundary length $|\partial B_{\text{local}}|$, since the delta function concentrates the integral on the line $\langle x - x_0, \nu \rangle = 0$. We want this local contribution to be

$$R_{local}(u) = d(i,j,\nu)\, |\partial B_{\text{local}}|. \tag{2.4}$$

Therefore we assume the constraints on $p$ in (2.3) and take the supremum over $p$.

This intuitive argument reveals that for the desired correct penalization with $d(i,j,\nu)$, when labels jump from $i$ to $j$ in direction $\nu$, we need the equality

$$\sup_{p \in C} \langle p_j - p_i, \nu \rangle = d(i,j,\nu). \tag{2.5}$$

From the definition (2.3) of the constraint set $C$, we can immediately conclude that at least $\leq$ holds. The question is now, what conditions must be imposed on $d$ to assure (2.5).

### 2.2. Assumptions on Direction Dependency of $d$

The inequality constraints in (2.3) can be written more concisely as

$$p_j - p_i \in C_{ij} \tag{2.6}$$

for all $i, j$, setting

$$C_{ij} := \left\{ z \in \mathbb{R}^m \,\middle|\, \langle z, \nu \rangle \leq d(i,j,\nu) \;\text{ for all } \nu \in \mathcal{S}^{m-1} \right\}. \tag{2.7}$$

This is a convex set, since every constraint is convex.

**Assumption** (Direction dependency)**.** We assume that $d$ is such that for every pair $i, j$, with the set $C_{ij}$ in (2.7):

$$d(i,j,\nu) = \sup_{z \in C_{ij}} \langle z, \nu \rangle. \tag{2.8}$$

The necessity, i.e. that (2.5) implies (2.8), follows from

$$\begin{aligned} d(i,j,\nu) &= \sup_{p \in C} \langle p_j - p_i, \nu \rangle \\ &\leq \sup_{p_j - p_i \in C_{ij}} \langle p_j - p_i, \nu \rangle \leq d(i,j,\nu). \end{aligned}$$

For a more intuitive formulation, in the following we extend $d(i,j,\cdot)$ positive homogeneously from $\mathcal{S}^{m-1}$ to whole $\mathbb{R}^m$, i.e. we set $d(i,j,t\nu) := t d(i,j,\nu)$ for all $t \geq 0$, $\nu \in \mathcal{S}^{m-1}$.

**Proposition 1.** *Equality* (2.8) *holds if and only if* $d(i,j,\cdot)$ *is convex, or equivalently if* $d$ *satisfies the triangle inequality*

$$d(i,j,z+w) \leq d(i,j,z) + d(i,j,w). \tag{2.9}$$

*Proof.* If (2.8) holds then $d(i,j,\cdot)$ is obviously convex. The converse is a basic property of support functionals [13]. □

### 2.3. Assumptions on Label Dependency of $d$

For the case that all jump directions are handled equally, i.e. $d(i,j,\nu) = d(i,j)$ for all $\nu \in \mathcal{S}^{m-1}$, the condition for (2.5) is that $d$ must satisfy the triangle inequality

$$d(i,j) \leq d(i,k) + d(k,j)$$

together with $d(i,i) = 0$ and $d(i,j) = d(j,i)$ [9]. For the general case, the condition is that this must hold for all $\nu$:

**Assumption** (Label dependency)**.** We assume that $d$ satisfies the triangle inequality

$$d(i,j,\nu) \leq d(i,k,\nu) + d(k,j,\nu) \tag{2.10}$$

together with $d(i,i,\nu) = 0$ and $d(i,j,\nu) = d(j,i,-\nu)$ for all $\nu \in \mathcal{S}^{m-1}$.

This is necessary, which follows easily from (2.5) by writing $p_j - p_i$ as $(p_j - p_k) + (p_k - p_i)$. The following proposition states that with (2.8) and (2.10) we indeed have found a *necessary and sufficient* condition for (2.5).

**Proposition 2.** *Equality* (2.5) *holds if and only if d satisfies assumptions* (2.8) *and* (2.10)*.*

*Proof.* See appendix. $\square$

This proposition gives an *exact characterization* of the distance functions expressible with our approach. The restrictions imposed are quite natural for distance functions. Note that the penalization is even allowed to be negative for some directions, meaning an endorsement of certain jumps.

## 3. Properties of the Regularizer

Based on proposition 2 we can prove the following main theorem of the paper. It shows that, assuming the necessary conditions (2.8) and (2.10), the regularizer (2.2) does indeed what it promises.

**Theorem 3.** *Let $d$ satisfy* (2.8) *and* (2.10)*. Let $u_i = \chi_A$, $u_j = \chi_{\bar{A}}$ with $A = \{x \in \Omega \,|\, \langle x - x_0, \nu \rangle \leq 0\}$ for some fixed $1 \leq i,j \leq n$, $\nu \in \mathcal{S}^{m-1}$ and $x_0 \in \Omega$. Then*

$$R(u) = d(i,j,\nu)\,\mathrm{Per}(A) \tag{3.1}$$

*where $\mathrm{Per}(A) = TV(\chi_A)$ is the perimeter of $A$ in $\Omega$.*

*Proof.* See appendix. $\square$

The theorem holds also in the general case as a global version of (2.4) with nearly the same proof: The local penalizations $d(i,j,\nu)$ are integrated over all jump interfaces weighted by the local interface length, yielding $R(u)$.

The regularizer $R(u)$ has the favorable property of being convex, rendering global optimization possible.

**Proposition 4.** *$R(u)$ is convex.*

*Proof.* For $0 \leq \alpha \leq 1$ and $u := \alpha u^1 + (1-\alpha)u^2$ we have

$$\begin{aligned} R(u) &= \sup_{p \in C} \Big( \alpha \int_\Omega \langle p, \nabla u^1 \rangle dx + (1-\alpha) \int_\Omega \langle p, \nabla u^2 \rangle dx \Big) \\ &\leq \alpha \sup_{p \in C} \int_\Omega \langle p, \nabla u^1 \rangle dx + (1-\alpha) \sup_{p \in C} \int_\Omega \langle p, \nabla u^2 \rangle dx \\ &= \alpha R(u^1) + (1-\alpha)\, R(u^2). \end{aligned}$$

$\square$

For the *continuous* label space, Alberti et al. [1, Lemma 3.7] give a relaxation of a general regularizer, which allows penalizations depending on the jump direction. Our regularizer 2.2 can be considered as its discretization to a finite number of labels. However, it is not clear if the direction dependency of [1] transfers correctly to the discrete setting. With theorem 3, we establish an exact result for arbitrary dimensions, showing that the specified penalties are indeed attained at given jump directions.

## 4. Constraint Sets

In this section we will give some examples of the constraint sets (2.7), arising by allowing labels to jump only in particular directions. The sets for common directions are depicted in Fig. 3. In general, $C_{ij}$ is the intersection of half-spaces, Fig. 3 (c–g), and can be found geometrically as in (f). In the case that $d$ is a seminorm, i.e. satisfies $d(\nu) \geq 0$ in addition to (2.9), the set is easily found to be the unit ball

$$C_{ij} = \{ w \in \mathbb{R}^m \,|\, d^*(w) \leq 1 \} \tag{4.1}$$

of the dual seminorm $d^*(w) := \sup_{z:\, d(z) \leq 1} \langle z, w \rangle$, Fig. 3 (a, b). An example of a general distance function is

$$d_A(\nu) = \begin{cases} \lambda & \text{if } \angle \nu \in A, \\ \infty & \text{else} \end{cases} \tag{4.2}$$

for some fixed $\lambda \geq 0$, which allows only jump directions with angles from a set $A$ and aside from that penalizes the interface length isotropically. Special cases are $d_{\{\alpha\}}$, $d_{\{\alpha,\alpha+\pi\}}$ and $d_{[\alpha,\beta]}$ for some angles $\alpha$ and $\beta$, Fig. 3 (a, c–f). These are already all cases due to (2.9). One can easily derive formulas for the projections onto these sets.

In discrete graph cut and dynamic programming approaches one only specifies $d(\nu)$ for the axis directions $\nu = \pm e_1$ and $\nu = \pm e_2$. The resulting set $C_{ij}$ is then a rectangle, Fig. 3 (b). Computing $d$ back by (2.8) e.g. for the square case, we see that this produces the penalization $d(\nu) = \|\nu\|_{l^1}$, which is not rotationally invariant as opposed to $\|\nu\|_{l^2}$, Fig. 3 (a). Thus, the discrete approaches form a subset of our framework, and we can actually *see* that they necessarily give rise to a metrication error.

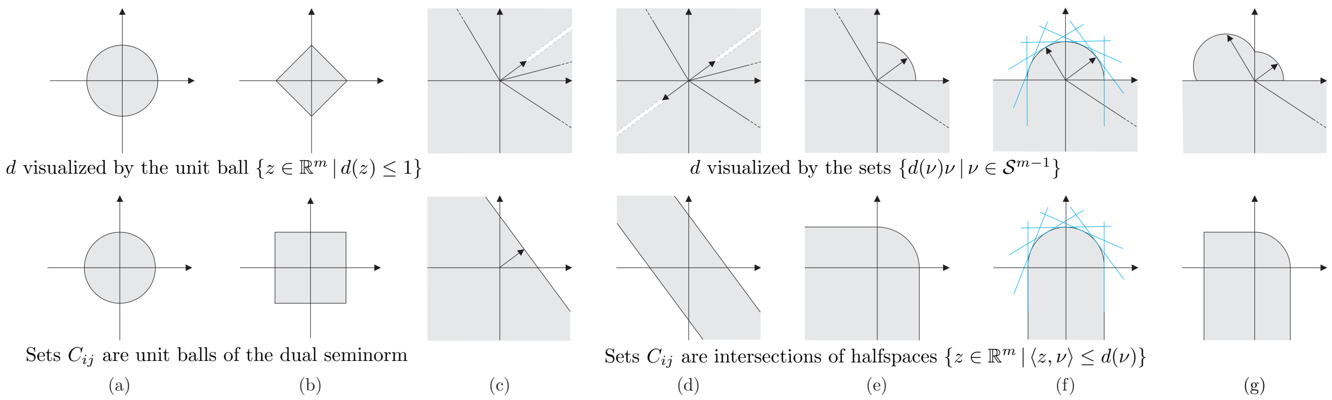
$d$ visualized by the unit ball $\{z \in \mathbb{R}^m \mid d(z) \le 1\}$
$d$ visualized by the sets $\{d(\nu)\nu \mid \nu \in \mathcal{S}^{m-1}\}$
Sets $C_{ij}$ are unit balls of the dual seminorm
Sets $C_{ij}$ are intersections of halfspaces $\{z \in \mathbb{R}^m \mid \langle z, \nu\rangle \le d(\nu)\}$
(a)
(b)
(c)
(d)
(e)
(f)
(g)

Figure 3. Different penalization functions $d$ (*top*) and the corresponding sets $C_{ij}$ in (2.7) (bottom). (a) and (b) show $d$'s arising by picking a specific seminorm, here the isotropic $\|\cdot\|_{l^2}$ and anisotropic $\|\cdot\|_{l^1}$, the $l^1$ norm as used in graph cuts. (c–g) show $d$'s arising by an explicit construction, following the arrows until the boundary gives the values $d(\nu)$. In (c) only one direction is allowed, in (d) only two, in (e) only right and up, in (g) only from bottom to top and in (f) just as in (g) but the left directions are penalized more.

## 5. Implementation

We solve the overall optimization problem (2.1), (2.2) using the fast primal-dual algorithm of [12]. It consists essentially in a gradient descent in $u$ and a gradient ascent in $p$, with reprojections onto the constraint sets.

The constraint $\sum_{i=1}^n u_i = 1$ on $u$ is implemented by Lagrange multipliers adding the terms $\sup_\sigma \sigma(\sum_{i=1}^n u_i - 1)$ to the energy. For the projection of $p$ onto the set $C$ in (2.3) we introduce auxiliary variables $q_{ij} = p_j - p_i$ and enforce these equalities by adding the corresponding Lagrange multiplier terms $\inf_{\lambda_{ij}} \sum_{ij} \langle \lambda_{ij}, p_j - p_i - q_{ij}\rangle$ to the energy. It remains then to project each $q_{ij}$ independently onto $C_{ij}$. In common cases, there is a simple formula for this. Otherwise, one can always use Lagrange multipliers for the constraints in $C_{ij}$. using some discrete range of directions $\nu$.

A prominent advantage of our method is that different ordering constraints are encoded only in the projections onto the constraint sets $C_{ij}$. The optimization algorithm itself remains the same. This is in contrast to [10] and [4], where the algorithms must be devised anew when the layout changes.

With a parallel CUDA implementation on NVIDIA GTX 480, usual runtimes for $640 \times 480$ images and 5 labels are around 90 seconds, and for $320 \times 240$ images 7 seconds. This compares favorably to the reported 9 seconds in [4].

## 6. Results

### 6.1. Geometric Class Labeling

In geometric class labeling, one seeks a rough labeling into geometric classes such as 'left wall' or 'sky'. This can be useful in providing geometric context to more elaborate tasks such as 3d reconstruction or robot navigation [5].

**Five regions layout.** Especially for indoor images, Liu et al. [10] used a simple layout consisting of five geometric parts: 'center' $C$, 'ceiling' $T$, 'floor' $B$, 'left wall' $L$ and 'right wall' $R$. Natural ordering constraints on these labels result in a layout as in Fig. 2. With the notation (4.2) we set

$$d(B,L) = d_{[\frac{\pi}{2},\pi]}, \;\; d(B,C) = d_{\{\frac{\pi}{2}\}}, \;\; d(B,R) = d_{[0,\frac{\pi}{2}]} \tag{6.1}$$

and likewise, accordingly rotated, for $L$, $R$ and $T$ instead of $B$. The distances for the remaining label pairs are then defined implicitly by the triangle inequality (2.10), and are found to be $d(L,R) = 2d_{\{0\}}$ and $d(B,T) = 2d_{\{\frac{\pi}{2}\}}$.

We applied our regularizer on the dataset of 300 indoor images from [10] using their dataterms. We set $\lambda = 20$ in (4.2) and weight all distances $d$ by $w(x) = e^{-|\nabla I(x)|^2/2\sigma^2}$ with $\sigma^2 =$ mean of $|\nabla I|^2$ for each input image $I$. Results for six different images are shown in Fig. 4. We achieved an overall accuracy of 85.3%, which is comparable to the 85% of [10]. The better result despite the fact that our method is a generalization of [10] may be due to several reasons: Our spatially continuous framework does not suffer from metrication errors. Moreover, convex optimization possibly finds better minima than iterative schemes like alpha-expansion and order-preserving moves.

**Tiered layout.** Tiered layout [4] is a generalization of the five region layout [10]. There are the 'top' $T$, 'bottom' $B$ and in between a number of labels, here $L$, $C$, $R$, in any sequence and multiplicity having only vertical borders between each other, Fig. 2. While the five regions layout assumes the center $C$ to be a rectangle, tiered layout is well adapted for outdoor images as well, where the boundary between $T$ and $L$, $C$, $R$ is less predictable, Fig. 7. We set

$$d(L,C) = d(C,R) = d_{\{0,\pi\}}, \; d(B,C) = d_{[0,\pi]},$$

$$d(B,R,\nu) = \begin{cases} d_{[0,\frac{\pi}{2}]}(\nu) & \text{if } \angle\nu \in (-\pi, \frac{\pi}{2}], \\ \lambda(2|\nu_1| + |\nu_2|) & \text{else,} \end{cases} \tag{6.2}$$

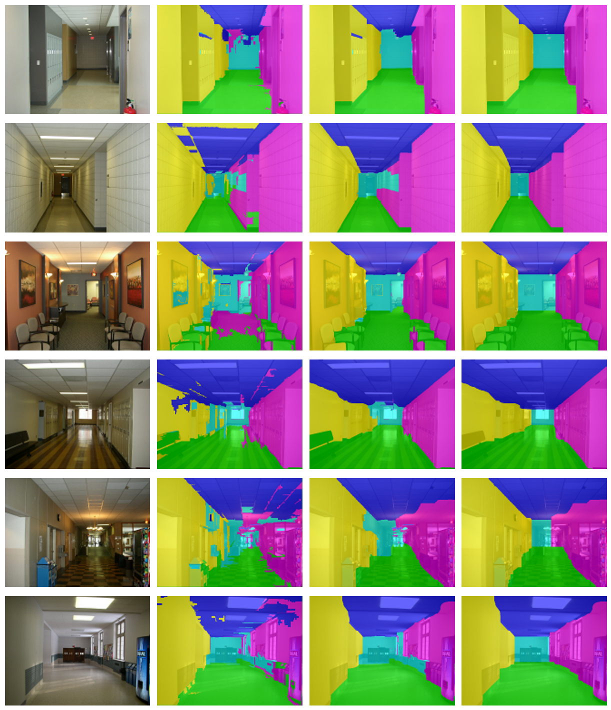
Figure 4. Five regions layout. Ordering constraints arise naturally for indoor images and improve the segmentation. *From left to right:* Indoor images, dataterm only labeling, result with Potts partitioning, result with five regions layout (Fig. 2).

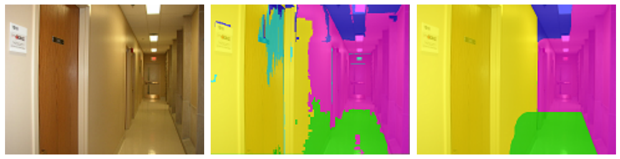
Figure 5. Failure case for the five regions layout. Since the dataterm (*center*) is very misleading, the ordering constraints do not allow to recover the desired solution. *From left to right:* input image, dataterm, result with ordering constraints.

and likewise for $d(B,L)$ and for $T$ instead of $B$. Other distances follow implicitly by (2.10), e.g. $d(L,R) = 2d_{\{0,\pi\}}$. In the optimization, only the projections for explicitly specified $d$'s must be taken into account. The expression for $d(B,R)$ results by combining the value $d_1 := d_{[0,\frac{\pi}{2}]}$ as in (6.1) and the estimate $d(B,R) \leq d(B,C) + d(C,R) = 2d_{\{0,\pi\}} =: d_2$ by (2.10). The corresponding set $C_{BR}$, Fig. 3 (g), is due to (2.7) the intersection of the sets $C_{d_1}$ and $C_{d_2}$, Fig. 3 (d, e). The value in (6.2) is then given by (2.8).

Six results on the dataset from [5], consisting of 300 outdoor images, are shown in Fig. 7. We used the confidence estimates provided in [5] for the dataterm and same parameters as for the five regions layout. Our overall accuracy 86.3% compares favorably to the 81.4% reported in [4].

Our method generalizes [4], in that we can handle all cases where the triangle inequalities are satisfied. While [4]

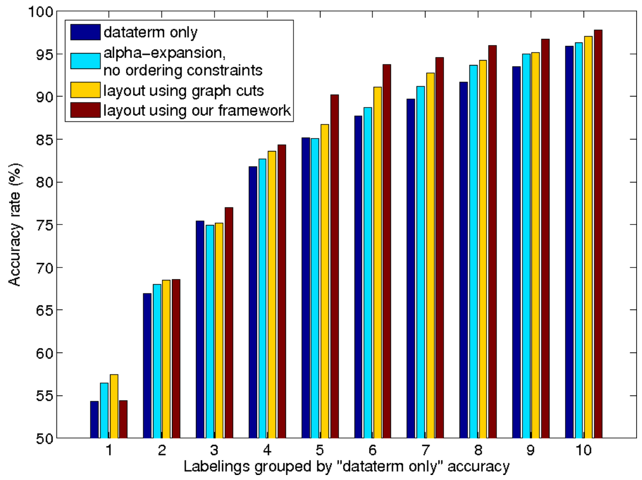

Figure 6. For the five regions model we ordered the 300 images by "dataterm only" accuracy in 10 equal groups. The proposed method provides a slight improvement over existing methods.

provides optimal solutions, it only applies to a very specific layout. Our method is capable of far more general layouts. This class is NP-complete containing the Potts model, so no globally optimal solutions can be expected. Nevertheless, our method provides tight a posteriori optimality bounds. The binary solution $u_i^{\text{bin}}(x) = 1,\ i := \arg\max_j u_j(x)$ is usually within 5% of the global optimum energy-wise.

**Three and more tiers.** To demonstrate the flexibility of our framework, we can easily model more than three tiers, Fig. 2 and 9. For four tiers, the middle tier is split up in two, containing regions $L_1, C_1, R_1$ and $L_2, C_2, R_2$, respectively. Label $B$ is allowed to change to $L_1, C_1, R_1$ just as before to $L, C, R$ in (6.2), and labels $L_i, C_i, R_i$ for $i = 1, 2$ can change to $T$ as $L, C, R$ before. Within one level, labels $L_i, C_i, R_i$ change as $L, C, R$ in (6.2). Finally, $L_1, C_1, R_1$ can change "one level up" to $L_2, C_2, R_2$ in any direction heading from bottom to top: $d(L_1, R_2) = d_{[0,\pi]}$ etc. Other distances follow implicitly by the triangle inequalities.

In contrast, the dynamic programming approach [4] does not allow an easy extension to four tiers, yielding a significantly more complex algorithm.

**Floating objects.** We can also allow arbitrary "floating" objects on top of the layout, Fig. 2 and 8. The dataset [5] contains also the classes 'porous' and 'solid'. However, they do not fit into the tiered layout, since their relation to other objects is less predictable. These classes are therefore neglected in [4]. In contrast, our framework allows to include these extra objects.

To include the 'solid' class $S$, we regard it as being "on top" of the tiered layout. We then split up the $L$ region into $L_S$ and $L_0$, meaning the parts that do or do not contain $S$,

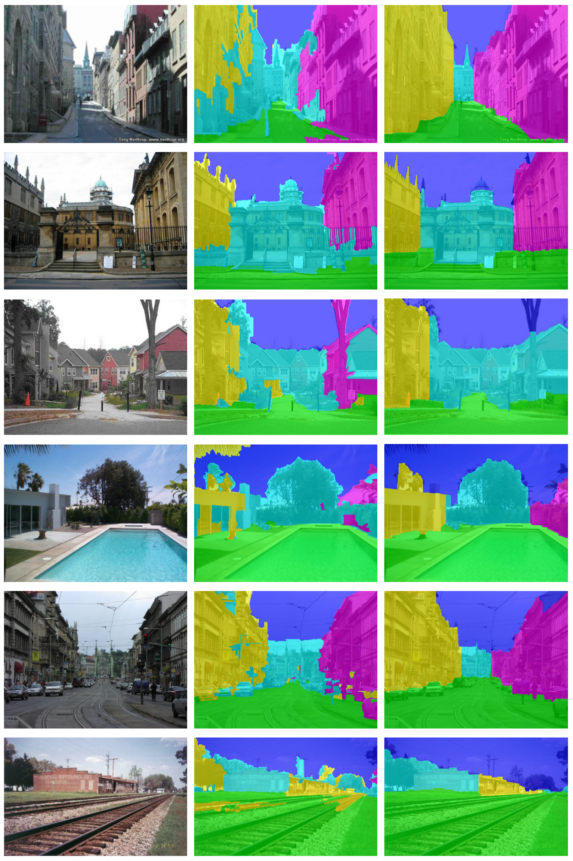

Figure 7. Tiered layout results. Ordering constraints improve the segmentation of outdoor images. *From left to right:* input images, Potts partitioning, partitioning with ordering constraints.

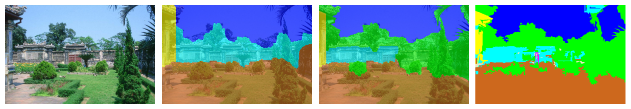

Figure 8. Our framework extends the tiered labeling to allow floating objects, arbitrarily located upon the layout. *From left to right:* input image, tiered layout result, result with extra objects 'solid' and 'porous', dataterm.

and similarly for other labels of the tiered layout. Jumps within non-$S$ labels, e.g. from $L_0$ to $C_0$ are then defined in the same way as previously from $L$ to $C$ by (6.2). Jumps within $S$-labels, e.g. from $L_S$ to $C_S$ only inherit the direction restrictions of $L$ and $C$, and the penalization is set to zero, since these are actually two parts of the same label $S$: Here we use (6.2) with parameter $\lambda = 0$ in (4.2). Finally, jumps from $S$-labels to non-$S$ labels just mean a jump from $S$ to some other label, so we can introduce an arbitrary new penalization here, e.g. the Potts model which penalizes all directions equally: $d(L_S, R_0) = d_{[0,2\pi]}$ etc.

Inclusion of two extra classes is done analogously, splitting each tiered layout label in three parts.

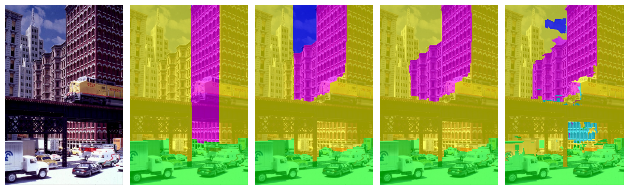

Figure 9. We can easily model the four and more tiered layout, thus allowing more than one label $L$, $C$ or $R$ to come one after another in one column. *From left to right:* input image, result with 3, 4 and 5 tiers, and dataterm.

## 6.2. Shape Priors

Using our approach we can model certain shape priors. The object of interest is divided into a number of sublabels, and some background labels are introduced surrounding this object. The idea is that changes from object labels to surrounding labels are constrained to certain jump directions only. We can model different shape priors expressible by the tiered labeling approach, such as a rectangle, trapezoid, 'house' or 'cross' [4]. The latter two are depicted in Fig. 10. However, our approach is more general, as it allows interfaces in arbitrary directions and not only vertical and horizontal. A simple example is the rotated cross prior in Fig. 10 which is not expressible using tiered layout. The distance functions $d$ can be easily derived from the layouts shown in Fig. 10.

**Star shape prior.** We note that the *star shape* prior [14] is expressible with our approach. This is an example where the distance function $d(x, i, j, \nu)$ also depends on the position $x \in \Omega$. Object and background are represented by the labels fg and bg, respectively. We fix an arbitrary point $x_0 \in \Omega$ and set

$$d(x, \mathrm{fg}, \mathrm{bg}, \nu) := \begin{cases} \lambda & \text{if } \langle x - x_0, \nu \rangle \geq 0, \\ \infty & \text{otherwise.} \end{cases} \tag{6.3}$$

for all $\nu \in \mathcal{S}^{m-1}$ with some $\lambda \geq 0$. This yields a $d$ as in Fig. 3 (f), with the "up" direction $x - x_0$. This way, the labeling is allowed to change to from object to background only in a direction "away" from the point $x_0$. Therefore, the object will be star shaped with center $x_0$.

**Novel prior: Convex shape.** A more advanced novel application of our framework is to model *convex priors*, i.e. to favor objects which are convex. To accomplish this, we explicitly model the boundary of the object. We introduce $n \geq 3$ auxiliary surrounding regions $0, \dots, n-1$, see Fig. 11. Let $\nu_i := (\cos \alpha_i, \sin \alpha_i)$ with $\alpha_i := i \cdot \frac{2\pi}{n}$. The main object label fg is then allowed to change to a label $0 \leq i \leq n-1$ only in direction $\nu_i$ and label $i$ to label $i+1$

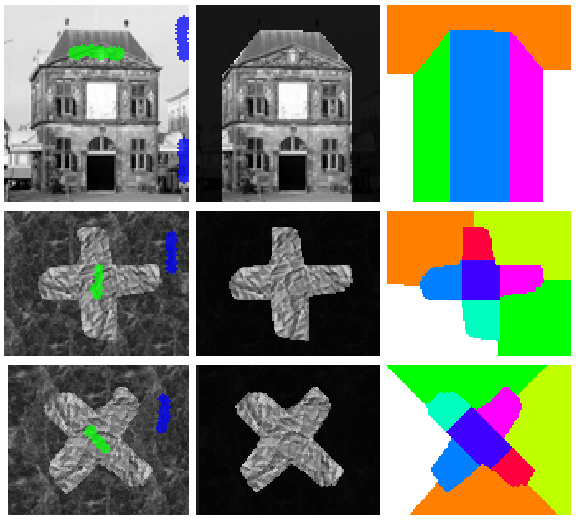
Figure 10. Segmentation with different shape priors. In contrast to [4], the proposed approach also allows rotated crosses. *From left to right:* initial image with user scribbles, segmentation using the prior, prior encoding as a label layout. We use a simple probability estimation from user scribbles to obtain the dataterm.

only in directions from $\nu_i^\perp$ to $\nu_{i+1}^\perp$:

$$d(\text{fg}, i) = d_{\{\alpha_i\}}, \quad d(i, i+1) = d_{[\alpha_i + \frac{\pi}{2}, \alpha_{i+1} + \frac{\pi}{2}]}, \qquad (6.4)$$

Fig. 3 (c, e). This way, the $n$ auxiliary regions are explicitly oriented around the object, building its boundary. Thus, we obtain a $n$-gon shape which is guaranteed to be convex.

## 7. Conclusion

We introduced a novel framework for a general multilabel regularizer allowing the penalization to depend on the jump direction. Although conceptually entirely different the proposed approach unifies and generalizes existing MRF-based formulations. Despite its generality, the regularizer is easily adapted to various label layouts by merely changing the projections of dual variables. In contrast, existing approaches require entirely different algorithms depending on the choice of layout. We proved a necessary and sufficient condition on the label distance function to be expressible with our framework. Quantitative experiments show that the proposed method compares favorably to existing approaches.

**Acknowledgements.** We thank Xiaoqing Liu and Olga Veksler for kindly providing us the dataterms for the five regions layout.

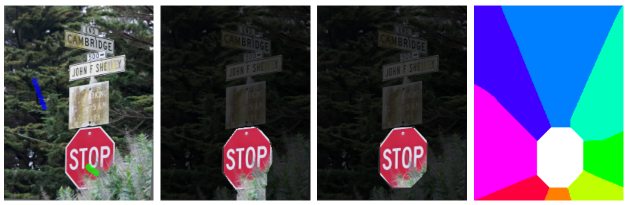
Figure 11. Beyond existing approaches, the proposed framework allows to model priors on convex shapes. *From left to right:* input image with user scribbles, Potts partitioning, partitioning using the convex shape prior, prior encoding. The prior is modeled by a $n$-gon, here an octagon, with edges pointing in all directions.

## References

[1] G. Alberti, G. Bouchitté, and G. D. Maso. The calibration method for the Mumford-Shah functional and free-discontinuity problems. *Calc. Var. Partial Differential Equations*, 16(3):299–333, 2003. 1, 3

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 23(11):1222–1239, 2001. 1

[3] A. Chambolle, D. Cremers, and T. Pock. A convex approach for computing minimal partitions. Technical report TR-2008-05, Dept. of Computer Science, University of Bonn, Bonn, Germany, November 2008. 1

[4] P. F. Felzenszwalb and O. Veksler. Tiered scene labeling with dynamic programming. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 0:3097–3104, 2010. 1, 4, 5, 6, 7

[5] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *Int. J. Comput. Vision*, 75:151–172, October 2007. 4, 5

[6] H. Ishikawa. Exact optimization for Markov random fields with convex priors. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 25(10):1333–1336, October 2003. 1

[7] N. Komodakis and G. Tziritas. Approximate labeling via graph-cuts based on linear programming. *IEEE Trans. on Patt. Anal. and Mach. Intell.*, 29(8):1436–1453, 2007. 1

[8] J. Lellman, J. Kappes, J. Yuan, F. Becker, and C. Schnörr. Convex multi-class image labeling by simplex-constrained total variation. Technical report, IPA, HCI, Dept. Of Mathematics and Computer Science, Univ. Heidelberg, October 2008. 1

[9] 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)*, pages 646 – 653, 2009. 3

[10] X. Liu, O. Veksler, and J. Samarabandu. Order-preserving moves for graph-cut-based optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1182–1196, 2010. 1, 4

[11] K. Murota. *Discrete Convex Analysis*. SIAM, 2003. 8

[12] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the piecewise smooth Mumford-Shah functional. In *IEEE International Conference on Computer Vision (ICCV)*, Kyoto, Japan, 2009. 4

[13] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1996. 2

[14] O. Veksler. Star shape prior for graph-cut image segmentation. In *Proceedings of the 10th European Conference on Computer Vision: Part III*, pages 454–467, Berlin, Heidelberg, 2008. Springer-Verlag. 1, 6

[15] C. Zach, D. Gallup, J.-M. Frahm, and M. Niethammer. Fast global labeling for real-time stereo using multiple plane sweeps. In *Workshop on Vision, Modeling and Visualization*, October 2008. 1

## A. Appendix

*Proof of theorem 3.* (2.2) gives integrating by parts:

$$\begin{aligned} R(u) &= \sup_{p\in C}\int_\Omega \sum_{k=1}^n u_k(-\operatorname{div} p_k)\\ &= \sup_{p\in C}\int_A(-\operatorname{div} p_i)+\int_{\bar A}(-\operatorname{div} p_j)\\ &= \sup_{p\in C}\int_{\partial A\cap\Omega}\langle -p_i,\nu_{\partial A}\rangle+\int_{\partial A\cap\Omega}\langle -p_j,\nu_{\partial\bar A}\rangle \end{aligned}$$

by divergence theorem. We have $\nu_{\partial A}=\nu$ and $\nu_{\partial\bar A}=-\nu$:

$$\begin{aligned} &= \sup_{p\in C}\int_{\partial A\cap\Omega}\langle p_j-p_i,\nu\rangle\\ &= \int_{\partial A\cap\Omega} d(i,j,\nu)=d(i,j,\nu)\operatorname{Per}(A). \qquad \square \end{aligned}$$

*Proof of proposition 2.* We introduce new variables $x_{kl}=p_l-p_k$ and enforce these equalities by Lagrange multipliers:

$$\begin{aligned} \sup_{p\in C}\langle p_j-p_i,\nu\rangle &= \sup_{\substack{x_{kl}\in C_{kl},\,p\\ x_{kl}=p_l-p_k}}\langle p_j-p_i,\nu\rangle\\ &= \sup_{x_{kl}\in C_{kl},\,p}\langle x_{ij},\nu\rangle-\sup_\lambda\sum_{kl}\langle\lambda_{kl},p_l-p_k-x_{kl}\rangle\\ &= \sup_{x_{kl}\in C_{kl}}\langle x_{ij},\nu\rangle+\inf_\lambda\sum_{kl}\langle\lambda_{kl},x_{kl}\rangle\\ &\quad+\sup_p\Big(-\sum_{kl}\langle\lambda_{kl},p_l-p_k\rangle\Big)\\ &= \sup_{x_{kl}\in C_{kl}}\langle x_{ij},\nu\rangle+\inf_\lambda\sum_{kl}\langle\lambda_{kl},x_{kl}\rangle \qquad (A.1) \end{aligned}$$

with $\lambda$ such that

$$\sum_k\lambda_{ks}=\sum_l\lambda_{sl} \qquad (A.2)$$

for all $1\le s\le n$. This constraint arises by evaluating the supremum over $p$.

For the sake of readability, in the following we proceed with the case $m=2$. The general case is handled in the same way. For an arbitrary $z\in\mathbb{R}^m$ such that $\nu,z$ are linearly independent, we can write $\lambda_{kl}=a_{kl}\nu+b_{kl}z$ for some $a_{kl},b_{kl}\in\mathbb{R}$. Then (A.2) is equivalent to the two independent constraints

$$\sum_k a_{ks}=\sum_l a_{sl}\quad\text{and}\quad\sum_k b_{ks}=\sum_l b_{sl} \qquad (A.3)$$

and the $\lambda$-infimum in (A.1) can be written as

$$\inf_\lambda\sum_{kl}\langle\lambda_{kl},x_{kl}\rangle=\inf_a\sum_{kl}a_{kl}\langle x_{kl},\nu\rangle+\inf_b\sum_{kl}b_{kl}\langle x_{kl},z\rangle.$$

The infimum over $a$ with the constraint (A.3) is equal to zero, if the full graph on vertices $\{1,\dots,n\}$ with edge weights $\langle x_{kl},\nu\rangle$ has no negative cycles (n.n.c.), and $-\infty$ otherwise [11, Proposition 5.1 and the proof], and similarly for the infimum over $b$. Thus, overall (A.1) leads to

$$\sup_{p\in C}\langle p_j-p_i,\nu\rangle=\sup_{\substack{x_{kl}\in C_{kl}\\ \langle x_{kl},\nu\rangle\text{ n.n.c.}\\ \langle x_{kl},z\rangle\text{ n.n.c.}}}\langle x_{ij},\nu\rangle \qquad (A.4)$$

By assumption (2.2), for every $k,l$ we can choose a $x^\nu_{kl}\in C_{kl}$ such that $\langle x^\nu_{kl},\nu\rangle=d(k,l,\nu)$ (respectively $\langle x^\nu_{kl},\nu\rangle=M$ with $M$ arbitrarily big in case $d(k,l,\nu)=\infty$). Since $d(\cdot,\cdot,\nu)$ satisfies the triangle inequality by assumption (2.10), all cycles with weights $\langle x^\nu_{kl},\nu\rangle$ are nonnegative (in case $d(k,l,\nu)=\infty$ for some $k,l$, choosing $M$ big enough makes the cycle nonnegative). In the following we will prove that we can choose a $z\ne\nu$ such that also $\langle x^\nu_{kl},z\rangle$ n.n.c. We then obtain, proving the proposition,

$$\sup_{p\in C}\langle p_j-p_i,\nu\rangle\ge\langle x^\nu_{ij},\nu\rangle=d(i,j,\nu)$$

(in case $d(i,j,\nu)=\infty$ we get l.h.s. $\ge M$ for any $M$ big enough, so again l.h.s. $=d(i,j,\nu)$).

Since the weights $\langle x^\nu_{kl},z\rangle$ depend continuously on $z\in\mathcal{S}^{m-1}$, cycles which are positive for $z=\nu$ remain positive for any $z\ne\nu$ sufficiently near $\nu$. One can easily see that this $\nu$-neighborhood can be chosen only depending on the sets $C_{kl}$ and $\nu$, and not on the particular choice of $x^\nu_{kl}$.

Now, consider a zero cycle $0=d(i_1,i_2,\nu)+\ldots+d(i_r,i_1,\nu)$. Fix an edge $(k,l)$ in the cycle. By triangle inequality this value is

$$\begin{aligned} 0&\ge d(k,l,\nu)+d(l,k,\nu)=d(k,l,\nu)+d(k,l,-\nu)\\ &=\sup_{x\in C_{kl}}\langle x,\nu\rangle-\inf_{x\in C_{kl}}\langle x,\nu\rangle\ge 0. \end{aligned}$$

So we actually have $\sup_{x\in C_{kl}}\langle x,\nu\rangle=\inf_{x\in C_{kl}}\langle x,\nu\rangle$, i.e. $\langle x,\nu\rangle=\text{const}=d(k,l,\nu)$ for all $x\in C_{kl}$. Therefore, for the $x^\nu_{kl}$ in the construction above we are free to choose *any* element of $C_{kl}$. In particular, we can choose $x^\nu_{kl}=x^z_{kl}$, so that $\langle x^\nu_{kl},z\rangle=\langle x^z_{kl},z\rangle=d(k,l,z)$ (or, analogously as above, $=M$ with $M$ arbitrarily big in case $d(k,l,z)=\infty$). But then the above $\nu$-zero cycle is $z$-nonnegative, since $d(\cdot,\cdot,z)$ satisfies the triangle inequality. So, all $z$-cycles are nonnegative too. $\square$