Cooperative Cuts for Image Segmentation

Stefanie Jegelka, Jeff Bilmes
jegelka@tuebingen.mpg.de, bilmes@ee.washington.edu

Max Planck Institute for Biol. Cybernetics
Tübingen, Germany

Dept of EE, University of Washington
Seattle WA, 98195-2500

UWEE Technical Report
Number UWEETR-2010-0003
August 2010

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Cooperative Cuts for Image Segmentation

Stefanie Jegelka, Jeff Bilmes
jegelka@tuebingen.mpg.de, bilmes@ee.washington.edu

Max Planck Institute for Biol. Cybernetics
Tübingen, Germany

Dept of EE, University of Washington
Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2010-0003
August 2010

Abstract

We propose a novel framework for graph-based cooperative regularization that uses submodular costs on graph edges. We introduce an efficient iterative algorithm to solve the resulting hard discrete optimization problem, and show that it has a guaranteed approximation factor. The edge-submodular formulation is amenable to the same extensions as standard graph cut approaches, and applicable to a range of problems.

We apply this method to the image segmentation problem. Specifically, Here, we apply it to introduce a discount for homogeneous boundaries in binary image segmentation on very difficult images, precisely, long thin objects and color and grayscale images with a shading gradient. The experiments show that significant portions of previously truncated objects are now preserved.

1 Introduction

Graph-cut approaches have become increasingly popular for image segmentation via energy minimization [1, 2, 3, 4, 5, 6, 7]. The segmentation problem is formulated as inference in a random field, and then solved as a minimum (s, t) cut in an appropriate graph (see Figure 2). In this paper, we focus on binary image segmentation where one aims to assign pixels to either the foreground or background, that is, in the graph cut, to either the s or the t segment, respectively. The graph captures the two terms of the energy (objective) function: (i) pixels should match the statistical properties of their segment, e.g., color or texture (terminal edges), and (ii) the boundary between object and background should be "smooth", that is, the total weight of the cut edges should be small (inter-pixel edges or pairwise potentials). This notion of smoothness enforces the foreground to be compact and connected, and the boundary to separate dissimilar pixels. We view this objective function as the sum of a loss function and a regularizer, where the loss function corresponds to the local information, and the regularizer is the cost of cutting the inter-pixel edges.

While this general approach often works well, there are some important cases where the cut-based smoothness criterion fails. Figure 1 shows examples from two classes. First, objects such as insects and trees do not have a short, but rather a very "non-smooth" boundary due to their long, fine parts. As a result, the smoothness term leads to excluding the legs for the sake of a compact object. This effect is called "shrinking bias" (e.g. [5]). If we decrease the influence of the smoothness term, more "leg" is included in the object but also much unwanted background clutter. Second, the edge weights increase with the similarity of the incident pixels and thus the graph-cut method fails to track boundaries in low-contrast regions, for example if parts of the object are shaded. The shaded regions are treated as one chunk as in Figure 1(b).

Since the objective is essentially a tradeoff between loss and regularizer, the two difficulties can partially be overcome by a better model for the preferences of a pixel towards foreground and background. Much research has been devoted to such models. Yet, grayscale images as in Figure 1(b) still pose a particular challenge, since the color information is lost. Other approaches to the shrinking bias are based on user interaction [5] or assumptions about the extensions of the object relative to a bounding box [8]. Still, consider a user interaction as in [5], where one must click on each extension: for objects such as trees, this becomes extremely cumbersome. Objects with fine holes such as the fan neither fit the clicking nor the solution with bounding boxes.

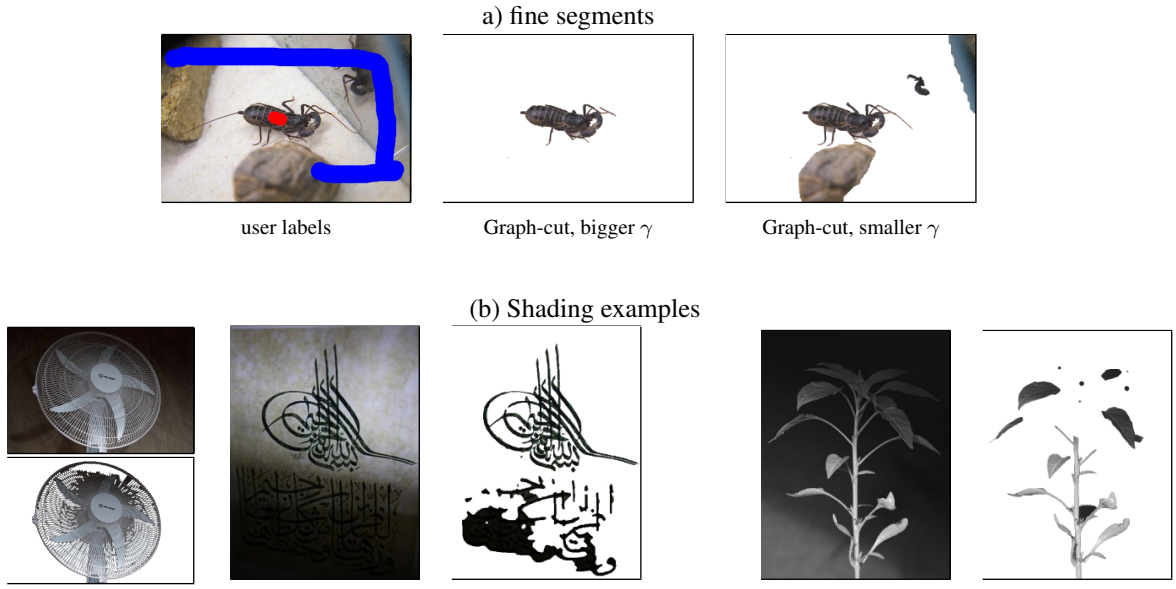


Figure 1: Examples of difficult segmentations where the standard smoothness assumption fails. a) Long segments and fine boundaries make the boundary overall long and "non-smooth". Lowering the coefficient γ of the smoothness term results in the inclusion of additional background objects, but the legs and tail are still not completely included. (b) Color (left) and grayscale (right) image with shading. In the color image, dark parts are treated as "chunks". In the grayscale image, the grayscale values do not provide enough information to the color model about what is foreground and background, so large parts are left out and others included. More examples are in Section 4. (Images are scaled down to reduce the file size of this report.)

In this report, we keep the loss term and specific user interaction aside and take a different direction: we modify the notion of "smoothness". What is an appropriate concept that captures cases as in Figure 1? We would like to retain a coherency effect, but at the same time make the true boundaries more favorable. A crucial observation is that these difficult boundaries are often still *homogeneous*. Along the boundary of the insect's legs in Figure 1(a), the color gradient remains uniform (brown to sand-colored) almost everywhere. In particular, it is similar around the "easier" body. Similarly, with a shading gradient as in Figure 1(b), both object and background become darker at roughly the same rate. Then, while linear pixel differences across the boundary of the object might change dramatically, the log-color gradient across that boundary, i.e., the ratio of pixel intensities, is stable with respect to shading.

We use this observation to extend the notion of "smoothness": a smooth boundary is either compact and traces high-contrast regions, or it is at least very uniform. That is, we favor homogeneous boundaries by granting them a selective discount. That way, a choice of boundary in the "easier" region can guide the choice of a boundary in the difficult, e.g., less contrastive, regions by discounting the weights of edges that are similar to the chosen clear boundary, and *only* of such edges. This homogeneity is invisible to the standard graph-cut objective with its linear sum of edge weights. We replace this *additive* regularizer by one that is *non-separable* across edges and can thus capture homogeneity across the entire image. Such global interaction usually comes at the price of intractability, except for special cases (e.g. [9]). Our regularizer is NP hard to minimize, but well approximable in practice.

To implement the discount for specific homogeneous groups of edges, which is essentially a *discrete structured regularization*, we choose a submodular cost function over edges. Submodular functions are an important concept in combinatorics [10], and also in (cooperative) game theory [11]. These functions are well suited to support coalitions thanks to their *diminishing marginal costs*[1]. We design a function on sets of edges that selectively diminishes the marginal costs within groups of similar edges, and behaves like the standard additive regularizer for sets of inhomogeneous edges. In the latter case, the boundary must satisfy the standard compactness requirement. The cost of the terminal edges, i.e., the "loss", remains untouched. Moreover, we allow a tradeoff between the traditional boundary smoothness as is desired by the graph-cut objective across all edges, and the homogeneity as desired by our modified

[1] In submodular games, the cost can be shared such that each individual is never worse off in a bigger coalition than in a smaller coalition that is a subset of the big one. Here, though, we do not implement any explicit cost sharing.

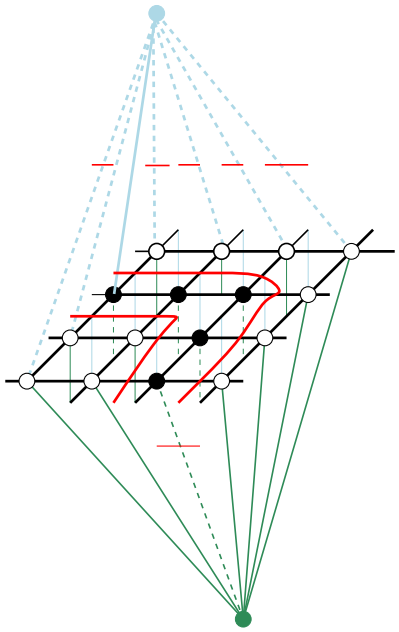


Figure 2: Image segmentation as a minimum (s, t) cut in a graph [1]. The image pixels form a grid of nodes, where each pixel is a node (black or white here) and each pixel is connected to its neighboring pixels. The weights of these (black) edges $\mathcal{E}_n$ depends on the similarity of the incident nodes. In addition, there are a source node s (blue) and sink node t (green) that are connected to each pixel. In the (s, t) cut (red), a pixel either remains on the source or the sink side; the source-side nodes will be the object, the sink-side nodes the background. In addition to the terminal edges, the cut must separate object (here black) and background (here white) by cutting all edges in between. This is only cheap if the boundary is short, or if the cut edges are "cheap", i.e., their joint weight is still low.

objective. We call a minimum cut with respect to the new objective a *Cooperative Cut* (CoopCut).

Submodular-cost cuts lead to a powerful class of objective functions that, as we shall see, are still approximable, and that when applied to the image segmentation yield good results on objects with homogeneous boundaries. Equally importantly, we provide a fast approximation algorithm (based on running the standard graph-cut algorithm multiple times). We also show that on objects that do not have homogeneous boundaries, Coopcut does not do worse than the standard approach. Since we merely replace the standard smoothness term and solve the cut by an iteration of min-cuts, the cooperative regularizer is compatible with many extensions to the standard model, such as user interaction [5] or additional (connectivity) constraints [12]. Furthermore, the algorithm can be easily parallelized on general purpose graphics-processing units (GPUs).

In the sequel, we first outline the standard approach and then detail our modification. Then we describe an efficient algorithm for finding an approximate CoopCut. Finally, we show the effectiveness of the cooperative regularization on images with shading, complicated boundaries and a standard benchmark.

1.1 Preliminaries, notation and definitions

A set function $f : 2^{\mathcal{E}} \to \mathbb{R}$ is said to be *submodular* [13, 14, 15] if it expresses diminishing marginal costs: for all $A \subseteq B \subset \mathcal{E}$ and $e \in \mathcal{E} \setminus B$, it holds that

$$f(B \cup \{e\}) - f(B) \leq f(A \cup \{e\}) - f(A),$$

that is, the marginal cost of e with respect to the larger set B is smaller than for A. A *modular* function satisfies this with equality and can be written as $f(A) = \sum_{e \in A} w_e$ where w_e is the cost of item $e \in \mathcal{E}$. A submodular function is *monotone* if $f(A) \leq f(B)$ for all $A \subseteq B$. Submodular functions are important in combinatorial optimization and economics, and are gaining attention in machine learning [16, 17, 18] and image segmentation [7]. The typical use in image segmentation, namely submodular potential functions, however, differs substantially with the use in this work (see also Section 2.3).

Consider a weighted undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ with $N = |\mathcal{V}|$ nodes $\mathcal{V}$, edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ and a modular weight function $w : \mathcal{E} \to \mathbb{R}_+$. Any edge $e = (u, v) \in \mathcal{E}$ corresponds to two nodes. A *cut* is a set of edges whose removal partitions $\mathcal{G}$ into at least two parts S and $\mathcal{V} \setminus S$. In reverse, any set of nodes $Y \subseteq \mathcal{V}$ determines a cut $\delta(Y) = \{(u, v) \in \mathcal{E} : u \in Y, v \in \mathcal{V} \setminus Y\}$ between Y and its complement $\mathcal{V} \setminus Y$. Note that in the sequel, a cut is a set of *edges*. Using $w(A) = \sum_{e \in A} w_e$, we define the graph cut function $\Gamma : 2^{\mathcal{V}} \to \mathbb{R}_+$ as the additive cost of the cut defined by Y:

$$\Gamma(Y) = w(\delta(Y)). \tag{1}$$

Several polynomial-time algorithms compute the minimizer $Y^* = \operatorname{argmin}_{Y \subset \mathcal{V}, Y \neq \emptyset} \Gamma(Y)$ of the *node* function $\Gamma(Y)$ via a minimum cut or, by duality, via a maximum flow [19, 2]. An (s, t) cut disconnects nodes s and t. An associated graph cut function on nodes is $\Gamma_{s,t} : 2^{\mathcal{V} \setminus \{s,t\}} \to \mathbb{R}_+$, $\Gamma_{s,t}(Y) = \Gamma(\{s\} \cup Y)$ for $Y \subseteq (\mathcal{V} \setminus \{s, t\})$.

In the sequel, $A, B, C, S \subseteq \mathcal{E}$ will denote sets of edges, and $Y \subseteq \mathcal{V}$ a set of nodes. The graph $\mathcal{G}$ will be the structure graph whose min-cut determines the MAP assignment of the associated random field $\mathfrak{G}$. Vectors $\mathbf{x}$ of pixels labels and $\mathbf{z}$ of pixel values are denoted in bold, with entries x_i and z_i. Vertices in the random field will also be denoted by x_i, $i = 1, \ldots, n$, for n pixels.

1.2 Image Segmentation by graph cuts

We start with a pixel-based viewpoint and from there pass over to the graph cut and edge-focused view which we eventually modify.

Binary image segmentation is often formulated in terms of a Markov random field (MRF) that represents a joint distribution $p(\mathbf{x}, \mathbf{z}; \xi)$ over labelings $\mathbf{x} = [x_1, \ldots, x_n]$ of the n pixels and over the observed image $\bar{\mathbf{z}}$, with parameters ξ. This $p(\mathbf{x}, \mathbf{z}; \xi)$ is a Gibbs distribution

$$p(\mathbf{x}, \bar{\mathbf{z}}, \xi) = Z^{-1} \exp(-E_\Psi(\mathbf{x}, \bar{\mathbf{z}}; \xi))$$

with normalizing constant (partition function) Z. The energy function $E_\Psi(\mathbf{x}, \bar{\mathbf{z}}; \xi)$ consists of potential functions,

$$E_\Psi(\mathbf{x}, \bar{\mathbf{z}}; \xi) = \sum_{T \in \mathcal{T}} \Psi_T(\mathbf{x}_T, \bar{\mathbf{z}}; \xi),$$

where $\mathcal{T}$ is the set of all cliques in the underlying graphical model $\mathfrak{G}$ of the random field. Finding a maximum a posteriori (MAP) labeling corresponds to finding $\mathbf{x}^* \in \operatorname{argmin} E_\Psi(\mathbf{x}, \bar{\mathbf{z}}; \xi)$ and is in general NP-hard.

If E_Ψ satisfies certain conditions (known as "regular" [2], and sometimes "attractive" [20] or "submodular" [21]; see also [22] and the earlier similar results in [23]; and [24, 25]), then the energy minimization can be solved exactly as a minimum cut in a particular weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ [1, 26]. For tractability and efficiency, the model is often restricted to local, *pairwise* MRFs, that is, the graphical model has a grid structure and Ψ_T is zero for all sets T with $|T| > 2$. The MAP assignment in $\mathfrak{G}$ then corresponds to a min-cut in the graph in Figure 2, with one node v_i for each pixel i. In addition, $\mathcal{G}$ has two terminals, s and t, corresponding to labels 0 (object) and 1 (background). The terminals are connected to each node v_i ($i = 1, \ldots, n$) by edges (s, v_i) and (v_i, t); these form the edge set $\mathcal{E}_t$. The edge (s, v_i) has weight $\Psi_i(x_i = 0, \mathbf{z}_b; \xi)$ and edge (v_i, t) weight $\Psi_i(x_i = 1, \mathbf{z}_b; \xi)$[2]. Thus, the terminal edges capture the unary potentials. These unary potentials express preferences for pixels to be either in the foreground or background based on their properties, e.g., color or texture. Furthermore, the edge set $\mathcal{E}_n = \mathcal{E} \setminus \mathcal{E}_t$ connects neighboring pixels in a grid structure. The weight of these inter-pixel edges corresponds to the pairwise potentials. An assignment $\mathbf{x}$ maps to an (s, t) cut in $\mathcal{G}$ in a straightforward way: Let $Y_\mathbf{x} = \{v_i | x_i = 0\}$. Then the cut is $\delta(Y_\mathbf{x})$, which we will also denote by $\delta(\mathbf{x})$. The edge weights in G are set in such a way that $E_\Psi(\mathbf{x}, \bar{\mathbf{z}}; \xi) + const = \Gamma_{s,t}(Y_\mathbf{x}) = w(\delta(Y_\mathbf{x} \cup \{s\}))$. The cut $\delta(\mathbf{x})$ consists of two subsets: the edges $\delta_n(\mathbf{x}) = \{e = (v_i, v_j) \in \mathcal{E}_n | x_i = 0, x_j = 1\}$ in $\mathcal{E}_n$ and the edges $\delta_t(\mathbf{x}) = \{e = (s, v_j) \in \mathcal{E}_t | x_j = 1\} \cup \{e = (t, v_j) \in \mathcal{E}_t | x_j = 0\}$ in $\mathcal{E}_t$.

Given the above analogy of potentials and cuts [1, 26], we define a cut-driven potential that is equivalent to E_Ψ: Let $C = \delta(Y)$ for some $Y \subset \mathcal{V} \setminus \{s, t\}$. Then

$$E_m(C) = \Gamma_{s,t}(Y) = \sum_{e \in C \cap \mathcal{E}_t} w_e + \gamma \sum_{e \in C \cap \mathcal{E}_n} w_e = w(C \cap \mathcal{E}_t) + \gamma w(C \cap \mathcal{E}_n). \tag{2}$$

The term on the right hand side extends to all sets $C \subseteq \mathcal{E}$, so we can view it as a set function on edges. The first term sums the weight of the cut $\delta_t(Y)$ of the terminal edges $\mathcal{E}_t$, and corresponds to the unary potentials; it is thus the data term. The second term is the cut $\delta_n(Y)$ through the inter-pixel edges; these edges enforce a coherent shape with a compact boundary. They correspond to the pairwise potentials, and their weights are proportional to the similarity of the adjacent pixels [1]. Finally, the parameter γ controls the tradeoff between the latter smoothness and the former data term. We will start our modification from the graph cut viewpoint in Equation (2).

The cut cost E_m in (2) is modular, and exactly this modularity of the coherency-enforcing cut through the grid $\mathcal{E}_n$ causes over-regularization which leads to the shrinking bias and shading effect in Figure 1. For long boundaries, $|C \cap \mathcal{E}_n|$ is large and so is the weighted sum, even for high-contrast edges. In low-contrast regions, the weight of the edges is so high that the sum grows large even for short boundaries. Hence, the additivity of the cut cost through the grid $\mathcal{E}_n$ is the subject to modify.

2 Structured regularization via Cooperative Costs

For a targeted discount, we replace the additive combination $w(C \cap \mathcal{E}_n)$ in E_m (Equation (2)) by a submodular cost.

More generally, let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$ where $f : 2^\mathcal{E} \to \mathbb{R}$ is a monotone, non-negative, and submodular set function defined on subsets of edges. This submodular cost yields a new, generalized cut function $\Gamma_f : 2^\mathcal{V} \to \mathbb{R}$, analogous to Equation (1):

$$\Gamma_f(Y) = f(\delta(Y)). \tag{3}$$

The function f is in general not additive any more, and can thus capture *global* interactions of edges, such as homogeneity. A CoopCut is the minimum cut with respect to cost f:

Definition 1 (Cooperative Cut)**.** *Given $\mathcal{G} = (\mathcal{V}, \mathcal{E}, f)$ with a submodular cost function $f : 2^\mathcal{E} \to \mathbb{R}$, find a(n (s, t)) cut $C \subseteq \mathcal{E}$ with minimum cost $f(C)$.*

The standard min-cut problem is a special case with $f = w$. We now re-define the cut potential using a submodular, cooperative cost over $\mathcal{E}_n$, while retaining the additive cost over $\mathcal{E}_t$:

$$E_c(C) \triangleq w(C \cap \mathcal{E}_t) + \gamma f(C \cap \mathcal{E}_n). \tag{4}$$

[2]The two edges can be summarized by one edge whose weight is the difference between the unary potentials for assignments one and zero.

Here, the structure of the graph is always the same grid as the min-cut graph for E_m, but any graph is conceivable. Note that f might itself be a mixture of a submodular function and the original modular function, thus allowing to tradeoff preference for homogeneity versus traditional graph-cut based smoothness. The weights $w(C \cap \mathcal{E}_t)$ can come from any unary potential model that is used in common graph cut methods. In fact, all weights w_e and f also depend on the observed image $\bar{\mathbf{z}}$, but we drop the subscript $\bar{\mathbf{z}}$ for notational convenience.

Back to MRFs, the cooperative cut potential (4) corresponds to a *global* MAP potential with respect to node labelings:

$$E_{\text{coop}}(\mathbf{x}, \mathbf{z}; \xi) = \sum_{v_i \in \mathcal{V}} \Psi_i(\mathbf{x}, \mathbf{z}; \xi) + \Psi_f(\mathbf{x}, \mathbf{z}; \xi) = \sum_{v_i \in \mathcal{V}} \Psi_i(\mathbf{x}, \mathbf{z}; \xi) + f(\delta_n(\mathbf{x}))$$

The potential is global because f is neither separable over edges nor nodes. Hence, the structure of the graphical model $\mathfrak{G}$ of the MRF is very different from that of the structural grid graph $\mathcal{G}$ of pixels.

2.1 Cooperative segmentation

The cooperative segmentation in Equation (4) is the skeleton to implement a discrete structured regularization, but needs to be fleshed with details. We wish to express an allowance for homogeneous boundaries without losing the intended smoothing effect of the pairwise potentials in the standard graph-cut approach. To implement this idea, we define classes of similar edges, $\mathcal{S}(\mathbf{z}) = \{S_1, S_2, \ldots, S_\ell\}$, $S_i \subseteq \mathcal{E}$ and $\mathcal{E} = \bigcup_i S_i$. Overlapping classes are possible but may require a re-weighting if not all edges are contained in the same number of classes, i.e., $|\{i | e \in S_i\}|$ differs across edges. The similarity can be (and usually is) a function of the observed image $\mathbf{z}$, and the resulting similarity classes are the groups that will enjoy a discount. In other words, we will target the property of diminishing costs to selectively reduce the cost of an additional edge e once we have cut enough edges from the same class as e, and the discount increases with the number of edges included from that class. That means within a class, the cost function f is submodular to reward homogeneity. Across different classes, f is modular to force an inhomogeneous boundary to be compact:

$$f(C) = \sum_{S \in \mathcal{S}(z)} f_S(C).$$

The functions f_S are monotone, normalized and submodular. It is well-known that a sum of submodular functions is submodular. In the language of games, any coalition $A = \cup_i S_i$ that is the union of several classes (e.g., $\mathcal{E}$), is "inessential" (if the S_i are disjoint); the cost f is only indecomposable for sets that are subsets of one class [27], that is, there is no gain to form coalitions across groups. The group costs f_S are thresholded discount functions of the form

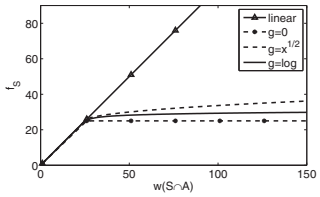
$$f_S(C) = \begin{cases} w(C \cap S) & \text{if } w(C \cap S) \leq \theta_S \\ \theta_S + g(w(C \cap S) - \theta_S) & \text{if } w(C \cap S) > \theta_S \end{cases} \tag{5}$$

for any nondecreasing, nonnegative concave function $g : \mathbb{R} \to \mathbb{R}$. For the experiments, we chose $g(x) = \sqrt{x}$ and $\theta_S = \vartheta w(S)$. The discount sets in only after a fraction ϑ of the weight of class S is contained in the cut. The parameter ϑ is a continuous shift between completely modular cuts ($\vartheta = 1$) and completely cooperative cuts ($\vartheta = 0$). Other conceivable functions g are the logarithm $g(x) = \log(1 + x)$, roots $g(x) = x^{1/p}$, and also convex mixtures with the modular non-discounted case $g(x) = x$. Figure 3(a) shows some examples. The submodularity of a thresholded discount function follows similarly to the submodularity of functions of the type $h(C) = g(|C|)$ for a concave, nondecreasing g [14], see also [27]. We defined the model is in this generality to show that it can be adopted to various settings.

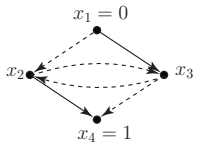
From the MRF viewpoint, in our model, all nodes adjacent to any edge in a particular class S form a clique in the underlying graphical model $\mathfrak{G}$, since $f(\delta_n(\mathbf{x})) = \sum_{S \in \mathcal{S}(z)} f_S(\delta_n(\mathbf{x}))$.

2.2 Edge classes

It remains to be determined what precisely "homogeneous" means in a given image. We infer the classes S via unsupervised learning, e.g., k-means clustering. The notion of homogeneity is then defined by the similarity measure. We use the ℓ_1 or ℓ_2 distance with respect to features $\phi(e)$, that are, for an edge $e = (v_i, v_j)$, based on the observed pixel values z_i, z_j. Two possibilities for features are the following:



(a) g functions



(b) graph

Figure 3: (a) Examples of g for use in Equation 5 and their effect. (b) Submodular edge functions do not guarantee node-submodular potential node functions. Let all edges in the graph belong to one class S, and $\Psi(x_1, x_2, x_3, x_4) = f_S(\delta_n(\mathbf{x})) = \sqrt{w(\delta_n(\mathbf{x}) \cap S)}$. Let the weights of the dashed edges be 0.1, and the weights of the other edges 9.9. We cut an edge if its tail has label zero and its head label one, and vary the labels of x_2 and x_3. Then $\Psi_S(0,0,1,1) + \Psi_S(0,1,0,1) = \sqrt{0.1 + 9.9 + 9.9} + \sqrt{0.1 + 0.1 + 0.1} < 5.01 < 6.32 < \sqrt{0.1 + 9.9} + \sqrt{0.1 + 9.9} = \Psi_S(0,0,0,1) + \Psi_S(0,1,1,1)$, violating regularity and node-submodularity. Combining such terms (if the given graph is a sub-graph of a bigger model) can lead to a global non-submodular energy function.

1. Linear color gradients: $\phi(e) = z_j - z_i$. This first-order difference can be substituted with linear least-squares slope estimates taken in the direction of a line defined by z_j, z_i.

2. Log ratios: To tackle shading problems, we compare the ratio of intensities (channel-wise for color images): $\phi(e) = \log(z_j / z_i)$.

2.3 Node-submodularity versus edge-submodularity

The main occurrence of submodularity in Computer Vision so far is as a property of potential functions that are defined on subsets of *nodes*, not edges. Importantly, the edge-modular graph cut function $\Gamma(Y)$ in Equation (1) is submodular on subsets of nodes for nonnegative edge weights. Conversely, submodularity-like properties such as "attractive" [20] or "regular" [7] imply that a potential function can be optimized as the min-cut in a graph. A potential function Ψ is *regular* if its projection onto any two variables is submodular: for an assignment $\mathbf{x}$, let $\Psi(y_i, y_j, \mathbf{x}_{V \setminus (x_i, x_j)})$ be the potential $\Psi(\tilde{\mathbf{x}})$, where $\tilde{x}_i = y_i$, $\tilde{x}_j = y_j$ and $\tilde{x}_k = x_k$ for all $k \neq i, j$. Then Ψ is regular if for any $\mathbf{x}$, it holds that

$$\Psi_V(0, 1, \mathbf{x}_{V \setminus (x_i, x_j)}) + \Psi_V(1, 0, \mathbf{x}_{V \setminus (x_i, x_j)}) \geq \Psi_V(0, 0, \mathbf{x}_{V \setminus (x_i, x_j)}) + \Psi_V(1, 1, \mathbf{x}_{V \setminus (x_i, x_j)}).$$

Figure 3(b) illustrates that cooperative, *edge*-submodular potentials do not in general satisfy this type of *node*-value-based submodularity. An analogous example shows that Γ_f from Equation (3) is not always a submodular function on nodes.

This difference between edge- and node-submodular potential functions is important. Indeed, edge-submodular potentials comprise edge-modular graph-cut potentials but are more powerful (see also Section 5.1). Edge-submodular potentials are in general NP hard to optimize, as opposed to node-submodular potentials [28].

Special cases of CoopCut that yield node-submodular potentials are (i) modular functions f_S, or (ii) each group S consists of all edges of a clique, and the edge weights of all edges in that clique are uniform [9].

3 Optimization Algorithm

As mentioned above and unlike edge-modular minimum cut, cooperative minimum cut is NP hard. In fact, no polynomial-time algorithm can guarantee a constant-factor approximation [28]. Hence, our approximation algorithm has a non-constant worst-case guarantee on the approximation factor. That said, the worst case is not necessarily the typical case, and such approximation algorithms often give solutions with good practical quality. The algorithm we describe here performed much better than its worst case bound on problems where the optimal solution was known [28].

Our algorithm iteratively minimizes an upper bound on the potential function. In such a setting, we aim for a bound that can be minimized efficiently. For graph cuts, a modular upper bound satisfies this criterion, since its minimization is merely a standard minimum cut. With such a bound, cooperative cut can be (approximately) solved in a practical amount of time. In addition, the algorithm is well suited for GPUs (using GPU implementations of max-flow [29]

and k-means [30]) and thus could be incorporated in a modern image-processing system. Furthermore, the algorithm can solve any graph cut with monotone submodular edge costs (as expressed in Equation 3), not only the model we propose here.

Is there such an "efficient" modular upper bound on f? The simplest upper bound on a submodular function f is the modular correspondent $f(A) \leq \hat{f}(A) = \sum_{e \in A} f(e)$. This bound is simple, but completely ignores the cooperation – and cooperation is a crucial ingredient of our model. Hence, we use a more informative bound that will include a restricted amount of cooperation. The restriction is with respect to a given set of edges $B \subseteq \mathcal{E}$; we call B the *reference set*. The *marginal cost* of an edge e with respect to B is commonly defined as $\rho_e(B) = f(B \cup \{e\}) - f(B)$, that is, the increase in cost f if edge e is added to $B \subset \mathcal{E}$. Now, given B, the following holds for any $A \subseteq \mathcal{E}$ [31]:

$$f(A) \;\leq\; h_{B,f}(A) \;\triangleq\; f(B) - \sum_{e \in B \setminus A} \rho_e(\mathcal{E} \setminus \{e\}) + \sum_{e \in A \setminus B} \rho_e(B).$$

Simply stated, $h_{B,f}(A)$ estimates $f(A)$ by subtracting the approximate cost of the elements in $B \setminus A$ and by adding the marginal cost of the elements in $A \setminus B$ with respect to B. The fact that $h_{B,f}$ is an upper bound follows from the submodularity of f, basically from diminishing marginal costs. If f is modular, then $\rho_e(B) = f(e)$ for any set B, and the upper bound is tight. It is always tight for $A = B$.

Let $\rho_e^S(B)$ be the marginal cost for f_S. Since the cost E_c is a sum of submodular functions, the potential is upper bounded by the sum

$$\begin{aligned} &E_c(A) \leq \\ &h_{B,E_c}(A) = E_c(B) - w(B \setminus A) + w(A \setminus B) - \gamma \sum_{e \in B \setminus A} \sum_{S \in \mathcal{S}(\mathbf{z})} \rho_e^S(\mathcal{E}_n \setminus \{e\}) + \gamma \sum_{e \in A \setminus B} \sum_{S \in \mathcal{S}(\mathbf{z})} \rho_e(B) \\ &\qquad = E_c(B) - w(B \setminus A) + w(A \setminus B) - \gamma \sum_{e \in B \setminus A} \sum_{S, e \in S} \rho_e^S(\mathcal{E}_n \setminus \{e\}) + \gamma \sum_{e \in A \setminus B} \sum_{S, e \in S} \rho_e(B). \end{aligned} \tag{6}$$

In the sequel, we will drop the subscript E_c for notational convenience.

Given an initial reference set B, we find the minimum cut with respect to h_B. Then we set $B = C$ and re-iterate until the solution no longer improves. Algorithm 1 shows the procedure. In the simplest case, we start with the empty set, $\mathcal{I} = \{\emptyset\}$. In fact, all results in Section 4 were obtained that way. This initial bound $h_\emptyset(A) = \hat{f}(A)$ is exactly the sum of edge weights, the cost of the standard graph cut.

The key observation to find the min-cut for h_B is that for a fixed reference set B, $f(B)$ is a constant and thus h_B becomes a modular function, and the optimization corresponds to a "standard" min-cut computation with edge weights

$$w_{h,B}(e) = \begin{cases} \rho_e(\mathcal{E} \setminus \{e\}) & \text{if } e \in B \\ \rho_e(B) & \text{otherwise.} \end{cases}$$

With these weights, the cost of a cut $A \subseteq \mathcal{E}$ is

$$\begin{aligned} \sum_{e \in A} w_{h,B}(e) &= \sum_{e \in B \cap A} \rho_e(\mathcal{E} \setminus \{e\}) + \sum_{e \in A \setminus B} \rho_e(B) \\ &= h_B(A) - \underbrace{f(B) + \sum_{e \in B} \rho_e(\mathcal{E} \setminus \{e\})}_{\text{constant w.r.t. } B}. \end{aligned}$$

Algorithm 1: Iterative bound minimization

Input: $G = (\mathcal{V}, \mathcal{E})$; nonnegative monotone cost function f: $2^{\mathcal{E}} \to \mathbb{R}_0^+$; reference initialization set $\mathcal{I} = \{I_1, \ldots, I_k\}$, $I_j \subseteq \mathcal{E}$; source / sink nodes s, t
Output: cut $B \subseteq E$
for $j = 1$ **to** k **do**
  set weights w_{h,I_j};
  find (s,t)-min-cut C for edge weights w_{h,I_j};
  repeat
    $B_j = C$;
    set weights w_{h,B_j};
    find (s,t)-min-cut C for edge weights w_{h,B_j};
  until $f(C) > f(B_j)$;
return $B = \arg\min_{B_1,\ldots,B_k} f(B_j)$;

For an efficient implementation, it is worthwhile to note that the marginal cost of an edge e only depends on the classes that contain e (Equation 6), that is, on the weights of e, of $(S_i \setminus \{e\})$ and of $(S_i \cap B) \setminus \{e\}$, and on the threshold θ_S.

The weights $w_{h,B}$ show how h_B differs from the simple $\hat{f}$ (where $w_{h,\emptyset} = f(e) = w_e$), and thus from the cost of the standard graph cut. The adaptive bound h_B comprises the cost-reducing effect of the edge classes with respect to B: If the current cut B contains a sufficiently large subset of edge class S, then $\rho_e(B) \ll f_S(e)$ for the remaining elements $e \in S \setminus B$. As an example, let $\theta_S = 20$ and $w(B \cap S) = 20$. If $w_e = 9$ for an $e \in S \setminus B$, then $f_S(e) = 9$, but $\rho_e(B) = 20 + \sqrt{9} - 20 = 3 < 9$. If the optimal cooperative cut A^* contains many edges, i.e., is non-smooth with respect to E_m, then its optimality must rely on exactly these discount effects captured by h_B.

3.1 Improvement via a Cut Basis

The initial $B = \emptyset$ leads to good solutions in many cases, as the experiments show. However, it can happen that the min-cut for $h_\emptyset$ includes unwanted edges, e.g., by including background noise into the object. If these unwanted edges come from big classes and the discount g becomes very flat, then their weight in the next iteration, $w_{h,B}(e) = \rho_e(\mathcal{E} \setminus \{e\})$, can be very small, and they remain in the cut. The initial cut biased us towards a local minimum that still includes the unwanted parts. This problem is solved by a larger set $\mathcal{I}$ of starting points that are sufficiently different, so that at least one does not include those edges.

We extend the set $\mathcal{I}$ of initial reference sets in a principled way. Partition the image into a graph G_{SP} of k superpixels [32] and build a spanning tree of G_{SP}. Each edge in the tree defines a cut: cutting the edge partitions the superpixels into two sets. We set $\mathcal{I}$ to those k cuts. Those cuts form a *cut basis* of all cuts of G_{SP}, i.e., a basis of the vector space of binary cut-incident vectors that indicate for each edge whether it is cut. The minimum cost cut basis (for a modular cost $h_\emptyset$) is defined by the Gomory-Hu tree [33, 34], but even a random spanning tree is suitable. Such a basis ensures that the set of initializations is widespread across the graph.

3.2 Approximation guarantee

The following Lemma gives an approximation bound for the initial solution for $B = \emptyset$, which can only improve in subsequent iterations.

Lemma 1. *Let $A_\emptyset = \operatorname{argmin}_{A \text{ a cut}} h_\emptyset(A)$ be the minimum cut for the cost $h_\emptyset$, and $A^* = \operatorname{argmin}_{A \text{ a cut}} E_c(A)$ the optimal solution. Let $\nu(A^*) = \min_{e \in A^*} \rho_e(A^* \setminus \{e\}) / \max_{e \in A^*} f(e)$. Then*

$$f(A_\emptyset) \leq \frac{|A^*|}{1 + (|A^*| - 1)\nu(A^*)} f(A^*).$$

Proof. We first use submodularity to bound $f(A_\emptyset)$ from above, with $e' = \arg\max_{e \in A^*} f(e)$:

$$f(A_\emptyset) \;\leq\; \sum_{e \in A_\emptyset} f(e) = h_\emptyset(A_\emptyset) \;\leq\; h_\emptyset(A^*) \;\leq\; |A^*| f(e'). \tag{7}$$

A lower bound on $f(A^*)$ is

$$\begin{aligned} f(A^*) &\geq f(e') + \sum_{e \in A^* \setminus \{e'\}} \rho_e(A^* \setminus \{e\}) \\ &\geq f(e') + (|A^*| - 1) \min_{e \in A^* \setminus \{e'\}} \rho_e(A^* \setminus \{e\}) \\ &\geq f(e') + (|A^*| - 1) \min_{e \in A^*} \rho_e(A^* \setminus \{e\}) \\ &= f(e') + (|A^*| - 1)\nu(A^*) f(e'). \end{aligned} \tag{8}$$

From (8), it follows that $f(e') \leq f(A^*)/(1 + (|A^*| - 1)\nu(A^*))$. This, together with (7), implies the approximation factor. □

For E_c with f_S as in Equation (5) and $g(x) = \sqrt{x}$, the term $\nu(A^*)$ is always nonzero. Its actual value depends on the cooperative cost reduction captured by ρ_e, i.e., the number of classes and the thresholds θ_S. If A^* has many more edges than $A_\emptyset$ (the cuts differ a lot), then $\rho_e(A^* \setminus e)$ must be small for most of the $e \in A^*$. Recall that Lemma 1 holds for $A_\emptyset$, and the solution improves in subsequent iterations. In an iteration with reference B, the weight of an edge e in $A^* \setminus B$ is $\rho_e(B) \leq \rho_e(A^* \cap B)$. If $B \cap A^*$ is large enough, then $\rho_e(A^* \cap B)$ will already be much smaller than $f(e)$, making the these edges much more preferable. This reduction is very likely if $|A^*|$ is large or if there are only few classes relative to the overall number of edges, e.g., we used about 10 classes for an image with 240.000 nodes in the experiments. The basis cuts ensure that the set of all cuts is well covered, so that we are very likely to find one solution that has a sufficiently high overlap with the classes that determine the optimality of A^*.

gray	GMM	hist.
GC	10.40	10.29
Coop, 5 cl.	4.86	4.94
Coop, 6 cl.	**3.84**	**3.81**

color	GMM	hist.
GC	2.72	3.71
Coop, 6 cl.	**1.28**	2.12
Coop, 10 cl.	1.29	**1.99**

Parameters, grayscale

	GMM		hist.	
	$10^3\vartheta$	γ	$10^3\vartheta$	γ
GC	-	1.2	-	1.3
Coop, 5 cl.	0.8	1.5	0.8	3.5
Coop, 6 cl.	1.0	1.5	0.8	3.5

Parameters, color

	GMM		hist.	
	$10^3\vartheta$	γ	$10^3\vartheta$	γ
GC	-	0.1	-	0.1
Coop, 6 cl.	2.0	0.5	2.0	0.7
Coop, 10 cl.	3.0	0.5	0.5	3.5

Table 1: Average percentage of wrongly assigned pixels (four images) for the shade benchmark. In particular for grayscale images, CoopCut yields better results.

4 Experiments

We compare CoopCut with a standard graph cut approach [1] (GC) on binary segmentation tasks on a public benchmark data set and our own, hand-labeled database of difficult settings. The experiments demonstrate that cooperative costs (i) improve over GC in tracking boundaries into shaded regions; (ii) yield better results than modular costs for objects with complicated boundaries; and (iii) do not harm the segmentation of objects that are neither slim nor shaded.

To ensure equivalent conditions, both methods receive the same 8-neighbor graph structure, terminal weights and inter-pixel weights w_e, so they only differ in their use of w_e. The weight of an inter-pixel edge $e = (v_i, v_j)$ is, in accordance with [5], $w_e = \lambda_1 + \lambda_2 \exp(-0.5\|z_i - z_j\|^2/\sigma)$, with similar parameters ($\lambda_1 = 2.5$ $\lambda_2 = 47.5$, and σ is the variance of the inter-pixel color differences). Since the comparison focuses on the smoothness cost and not the terminal edges, we use two simple standard color models, either histogram-based log probabilities as in [1] or a GMM with 5 components as in [3, 5]. Since CoopCut is complementary to the color model, any better color model can only improve the results.

The edge classes were generated by k-means clustering with ℓ_1 distance for the log ratio features (Exp. 1) and squared Euclidean distance for linear gradients (Exp. 2). Edges between identically colored pixels retained a modular cost ($\theta_S = w(S)$). Both CoopCut and GC were implemented in C++, using a graph cut library [2] and OpenCV, with some pre-processing in Matlab.

4.1 Experiment 1: Shading gradient

The first experiment addresses the shading problem. We collected 14 color and grayscale images of shaded objects; the lack of color information makes segmentation of grayscale images even harder. Here, we use CoopCut with the log ratio classes.

Figure 4 and 5 display how the grouping across light and darkness helps to find the "correct" boundaries; the "good" boundary indeed acts as a guide: whilst GC treats the dark area as one chunk, CoopCut captures many more details. This is even more pronounced for grayscale images, where the color model is less informative. Table 1 shows average quantitative errors for both methods on four hand-labeled grayscale and four color images (the mouse, the plants, and the fan). We picked the best parameters for either method. In particular on grayscale images, cooperative costs reduce the error by $> 50\%$.

4.2 Experiment 2: thin, long objects and complicated boundaries

To test the effect of the cooperative smoothness term for objects with long, slim parts, we collected a "leg benchmark" of 20 images with such objects. For each image, we hand-labeled some object and background pixels for the color model; Figure 1 shows an example. Table 2 lists the percentage of wrongly assigned pixels averaged over 8 hand-labeled images; CoopCut achieves less than half the error of GC. Since the fine parts often only comprise a small part of the object, we additionally computed the error only on those difficult regions ("leg error") – Figure 9 shows two example "true" labelings in the reduced count. Here as well, cooperative weights reduce the error by about 40%.

The visual results in Figures 6, 7, 8 and 9 illustrate the improvement by CoopCut in recovering the correct shape. In particular, CoopCut accurately cuts out the "holes" in grids and between fine twigs. For the insect in Figure 1,

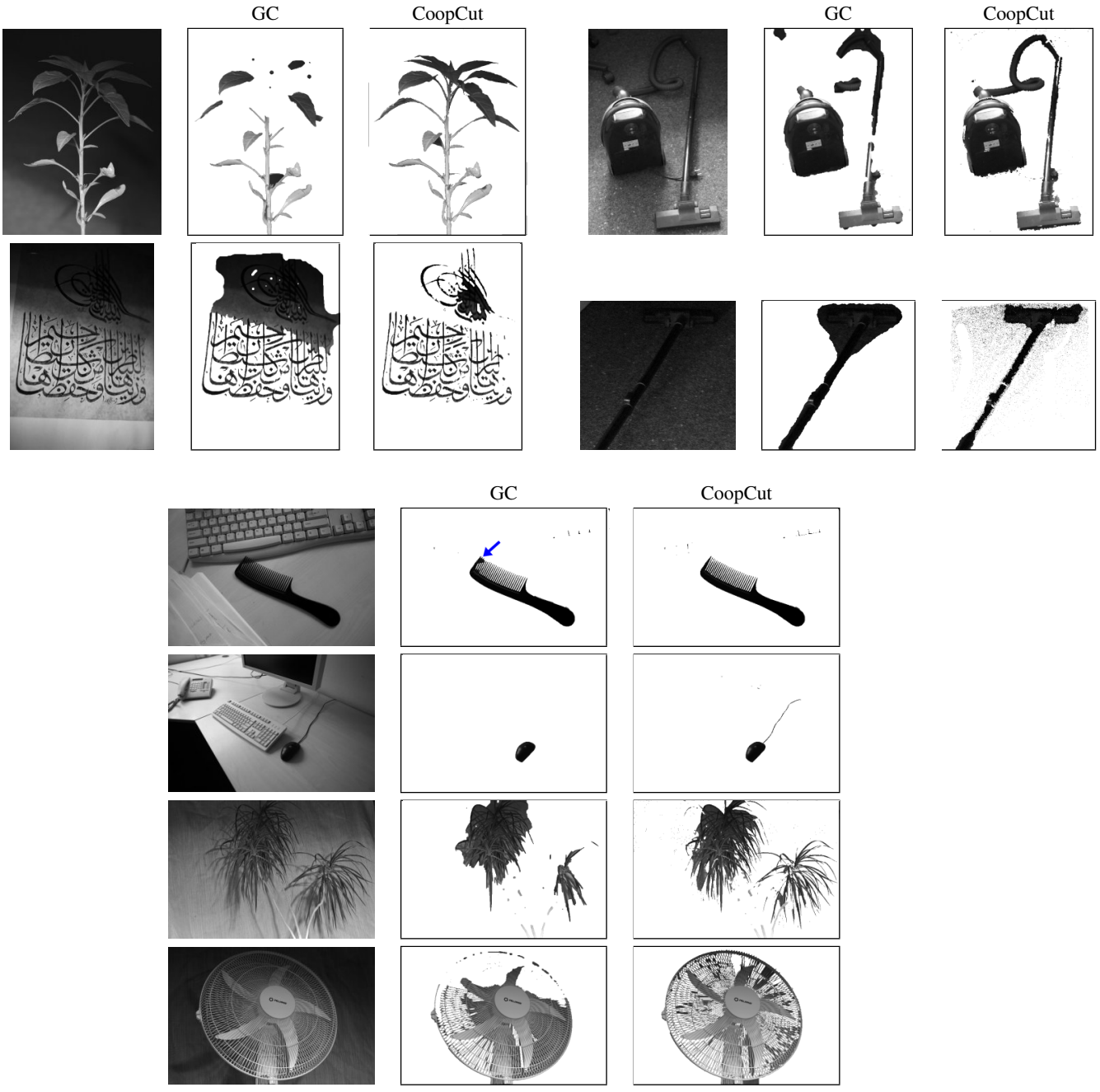


Figure 4: Best results for GC and CoopCut (6 classes) on the grayscale shading benchmark with the GMM color model (Images scaled).

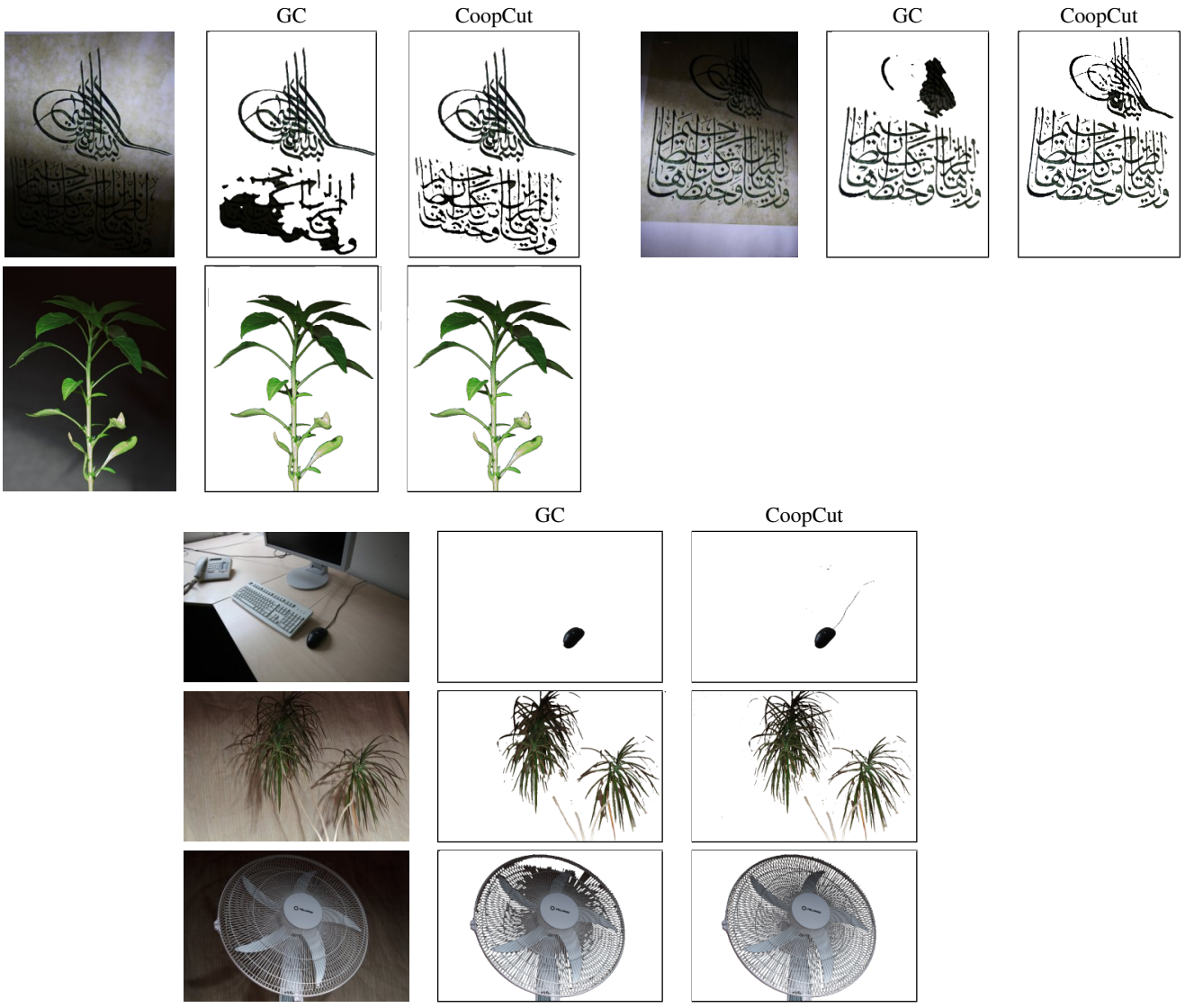


Figure 5: Best results for GC and CoopCut (10 classes) on the color shading Benchmark with the GMM color model (images scaled). For the palmtree-like plant, note the difference in the "holes" between the leaves (zoom in). Similarly, GC does not separate out all wires of the fan.

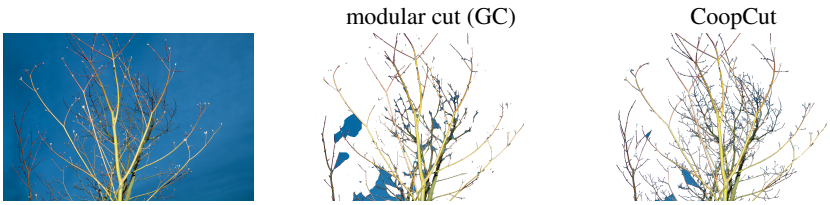


Figure 6: Example where the histogram color model leads to a poor segmentation (compare the same image with the GMM model in Figure 8): twigs are left out, but sky is included. Cooperative weights remedy large parts of the error.

	GMM				histogram			
	total	leg	total	leg	total	leg	total	leg
GC	2.43	35.89	4.74	18.42	2.70	52.13	5.28	32.03
Coop, 10 classes	0.90	11.65	**0.91**	**11.40**	1.16	27.42	**1.70**	20.65
Coop, 14 classes	**0.80**	20.13	0.94	12.33	**1.05**	26.61	1.76	**19.27**

Parameters

	GMM				histogram			
	$10^3\vartheta$	γ	$10^3\vartheta$	γ	$10^3\vartheta$	γ	$10^3\vartheta$	γ
GC	-	1.5	-	0.1	-	1.3	-	0.2
Coop, 10 classes	0.8	1.5	0.5	1.5	0.5	3.0	1.0	1.3
Coop, 14 classes	1.0	1.6	0.2	1.5	1.0	2.5	0.18	1.7

Table 2: Average error (%) on 8 images from the leg benchmark; for the parameters with the minimum error (left columns) and minimum joint error (right columns; leg plus 2 times total). Cooperative Cuts reduce the error by up to a half.

reducing γ in the standard model results only in the inclusion of background objects, but the fine boundaries are still spared. CoopCut, however, recovers most of the insect without background (Figure 7 top).

4.3 Experiment 3: Standard benchmark images

The final experiment addresses the loss incurred by the cooperative discount of weights on "standard" segmentation tasks with objects that are rounder and do not particularly fit the assumption of a "homogeneous boundary". These objects often require good color models and strong regularization; in short, they match the standard graph cut model well.

Table 3 displays the errors for both methods on the 50 images of the GrabCut data set [3, 35] with the "Lasso" labeling. Even here, CoopCut slightly improves the results on both color models, but less than for the difficult images above. The GrabCut data set includes two images where the standard method is known to face the shrinking bias: the tail of the cat is usually cut off (no. 326038, e.g. [35], Fig. 1b), as is the trunk of the "bush", unless lots of background is included (e.g. [8], Fig. 4). Figure 10 shows that CoopCut with the parameters in Table 3 preserves the trunk and tail, respectively. Admittedly, on some of the GrabCut images with very noisy background, GC yields clearer boundaries. On the other hand, the boundaries by GC can distort the shape tremendously, as with the cat or bush.

4.4 Summary and parameters

In summary, the experiments show a notable improvement for segmentations in difficult settings, while the segmentation in standard settings is not impaired. The results are usually stable over a range of parameters. Good parameters range from $\theta = 0.003w(S)$ to $0.0008w(S)$ and $\gamma = 0.8$ to $\gamma = 2.5$. In general, 10 classes is a good choice, and 6 classes for grayscale images. The optimal parameter choice varies slightly with the setting, as for standard graph cuts, but the errors show that one choice is reasonable for a wide range of images. Since the CoopCut improves on error rates for both color models, we expect that it equally does so for more sophisticated ones.

Figure 7: Best results for GC and CoopCut (10 classes) on the “Leg” Benchmark with the GMM color model, part I. Images were scaled after segmentation.

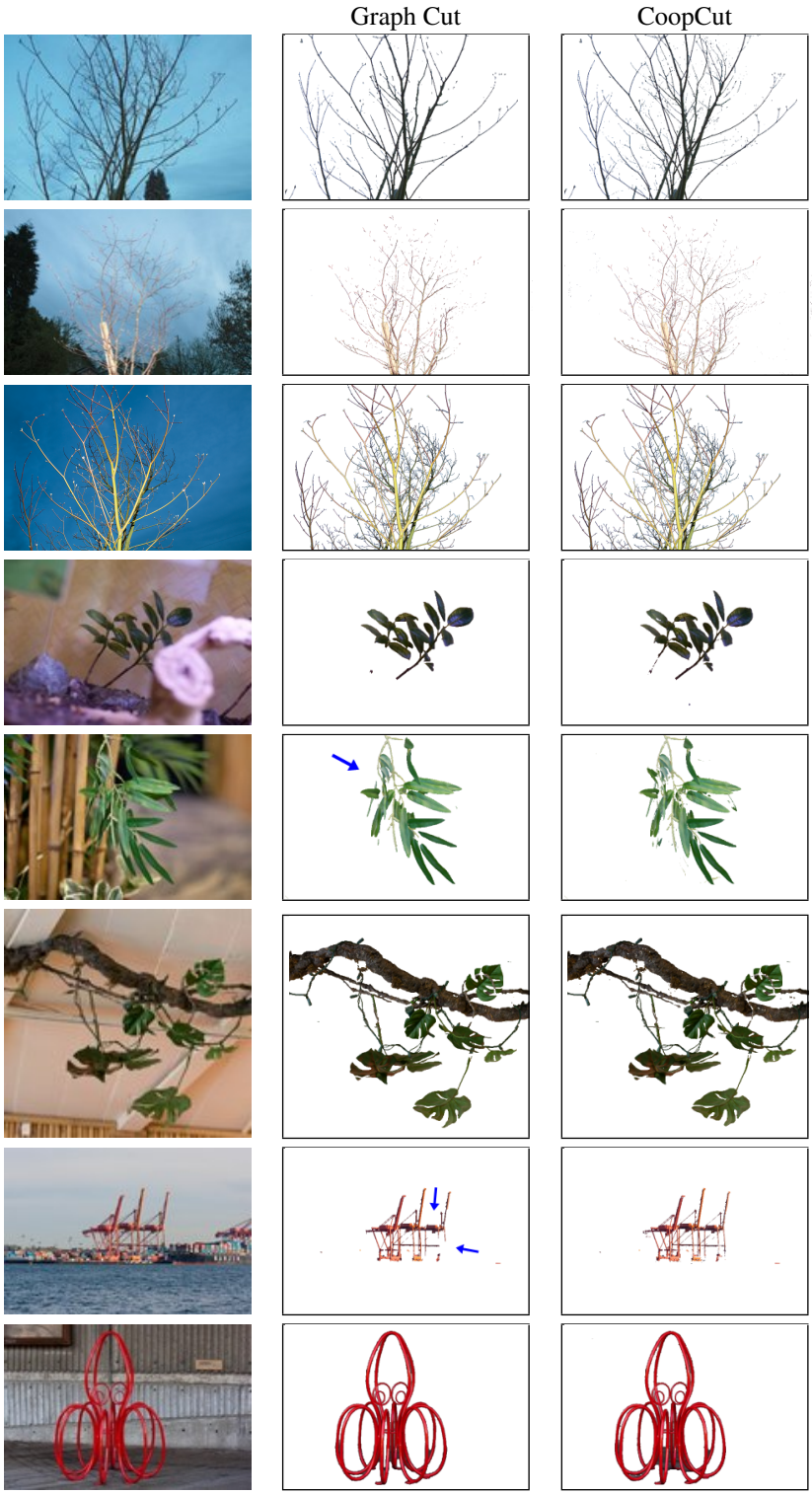


Figure 8: Best results for GC and CoopCut (10 classes) on the “Leg” Benchmark with the GMM color model, part II. For the trees, note the density of small branches. Images were scaled after segmentation.

Figure 9: Best results for GC and CoopCut (10 classes) on the "Leg" Benchmark with the GMM color model, part III. Images were scaled after segmentation. The grayscale images show examples of templates for the "leg error", where the gray pixels are ignored.

	GMM	hist.
GC	5.33 ± 3.7	6.88 ± 5.0
Coop Lin, 8 cl.	5.30 ± 3.7	6.79 ± 5.1
Coop Lin, 10 cl.	5.19 ± 3.5	6.74 ± 4.9
Coop Lin, 12 cl.	5.22 ± 3.8	6.71 ± 4.7

	GMM		hist.	
Parameters	$10^3\vartheta$	γ	$10^3\vartheta$	γ
GC	-	0.8	-	
Coop Lin., 8 cl.	1.0	0.8	7.0	0.6
Coop Lin., 10 cl.	1.0	0.8	3.0	1.5
Coop Lin., 12 cl.	1.0	1.2	1.0	2.5

	GMM	hist.
GC	5.33 ± 3.7	6.88 ± 5.0
Coop Ratio, 5 cl.	5.30 ± 3.7	6.86 ± 4.9
Coop Ratio, 8 cl.	5.25 ± 3.8	6.59 ± 4.5
Coop Ratio, 10 cl.	5.28 ± 3.7	6.50 ± 4.3

	GMM		hist.	
Parameters	$10^3\vartheta$	γ	$10^3\vartheta$	γ
GC	-	0.8	-	
Coop Rat., 5 cl.	1.0	1.2	5.0	0.5
Coop Rat., 8 cl.	1.0	1.2	0.2	10.0
Coop Rat., 10 cl.	3.0	0.5	0.2	10.0

Table 3: Average percentage of mispredicted pixels on the GrabCut data for CoopCut with linear gradient classes (left) and log ratio classes (right), compared to GC. We picked optimal parameters for each method.

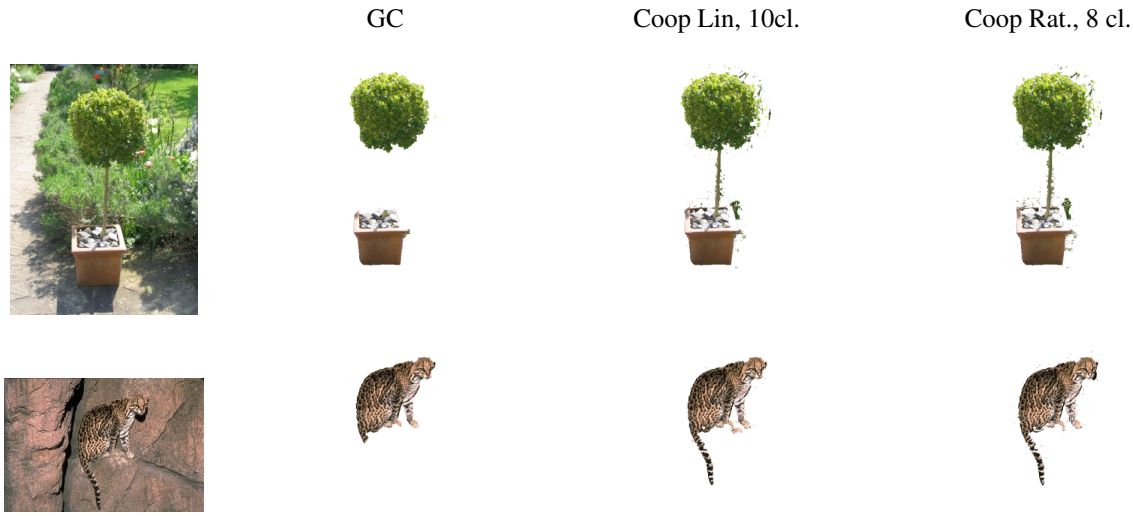


Figure 10: Examples of shrinking bias from the GrabCut data, results are with the parameters shown in Table 3.

5 Discussion

The cooperative cut functions we apply here are only few examples of the versatility of cooperative edge costs for graph cuts. As another example, we reduce some recently introduced higher order *node* potentials in a binary setting to CoopCut potentials.

5.1 Some higher-order node potentials as cooperative edge potentials

We show a reduction for P^n functions [9], P^n Potts potentials and robust P^n potentials [36] in the binary setting. In this case, both can be optimized exactly and then, an exact algorithm is of course better to use. Nevertheless, modeling these potentials as cooperative cuts shows the power of the latter and allows to combine them with other cooperative costs. In reverse, not all CoopCuts can be modeled by the given higher-order potentials. As an example, P^n functions lead to submodular node potentials, but cooperative cuts in general do not.

5.1.1 P^n Potts model

The P^n Potts potential is a generalization of the pairwise Potts potential and introduced in [9]. The P^n potential of a clique $T \subset V$ is

$$\Psi_T(\mathbf{x}) = \begin{cases} a & \text{if } x_i = x_j \; \forall \; x_i, x_j \in T \\ b & \text{otherwise,} \end{cases}$$

for $b > a$. To model this as a CoopCut in $\mathcal{G}$, create edges between all nodes in T, and set $w_e = b - a > 0$ for each such inter-clique edge e. Create a group $S \subseteq E$ that contains all clique edges, and set the submodular edge cost to $f_S(C) = \max\{w(C \cap S), b - a\} + a$. A shift to normalize f_S will not change the minimizing set.

5.1.2 P^n functions

(author?) [9] introduce a general familiy of potential functions for which move-making algorithms can be applied. These potentials are of the form[3]

$$\Psi_T(\mathbf{x}_T) = f_T\Big(\sum_{i,j \in T} \phi_T(x_i, x_j)\Big), \tag{9}$$

where $f_T : \mathbb{R} \to \mathbb{R}$ is concave non-decreasing and ϕ_T is a symmetric pairwise potential satisfying $\phi_T(1,1) \leq \phi_T(0,1)$ and $\phi_T(0,0) \leq \phi_T(0,1)$. Assume $\phi_T(1,1) = \phi_T(0,0)$. An equivalent subgraph with cooperative edge weights is the

[3]Here we assume that pair i, j and j, i are both counted, but the transfer to unsorted pairs is simple.

following: connect all nodes in clique T as a directed clique in the structure graph, and create a group S that contains these edges. Let now $w_e = 2(\phi(0,1) - \phi(0,0))$ for each $e \in S$ and

$$f_S(C) = f_T\Big(2|T|\phi_T(0,0) + \sum_{e \in C \cap S} w_e\Big).$$

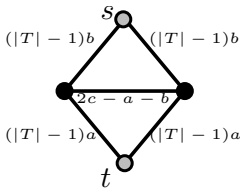


Figure 11: Graph for $a = \phi(0,0) < \phi(1,1) = b < \phi(0,1) = c$.

The shifted f_T is still concave non-decreasing and hence f_S submodular. The equality $f_S(\delta(\mathbf{x}_T)) = \Psi_T(\mathbf{x}_T)$ follows when equating each $\phi_T(x_i, x_j)$ with edge (v_i, v_j). Edge (v_i, v_j) is cut if $x_i = 0$ and $x_j = 1$.

If $\phi_T(0,0) \neq \phi_T(1,1)$, then we introduce interacting terminal edges. Let, without loss of generality, $a = \phi_T(0,0) < \phi_T(1,1) = b < \phi_T(0,1) = c$. For each node $v_i \in T$, we add the terminal edges (s, v_i) and (v_i, t) with weights $(|T| - 1)b$ and $(|T| - 1)a$, respectively, to S. That means the terminal weights have one "unit" for each possible pairing of the variable. Edges within the clique have weight $2c - a - b$ (see Figure 11). We set $f_S(\mathbf{x}_T) = f_T(w(\delta(\mathbf{x}_T))$. Let, for this section, $\mathcal{E}_t$ be the edges connected to t, and $\mathcal{E}_s$ the edges connected to s, and $C = \delta(\mathbf{x}_T)$. Then

$$\begin{aligned}
\gamma_{f_S}(\delta(\mathbf{x}_T)) &= f_T\big((|T|-1)(|C \cap \mathcal{E}_t|b + |C \cap \mathcal{E}_t|a) + |C \cap \mathcal{E}_n|(2c - a - b)\big) \\
&= f_T\Big(\sum_{i \in T, x_i = 0} a(|T|-1) + \sum_{j \in T, x_j = 1} b(|T|-1) + \sum_{i,j \in T, x_i = 0, x_j = 1} (2c - a - b)\Big) \\
&= f_T\Big(\sum_{i \in T, x_i = 0} \sum_{j \in T, x_j = 0} a + \sum_{i \in T, x_i = 0} \sum_{j \in T, x_j = 1} (a + 2c - a - b) + \sum_{i \in T, x_i = 1} \sum_{j \in T, x_j = 1} b + \sum_{i \in T, x_i = 1} \sum_{j \in T, x_j = 0} b\Big) \\
&= f_T\Big(\sum_{i,j \in T, x_i = x_j = 0} a + \sum_{i,j \in T, x_i = x_j = 1} b + \sum_{i,j \in T, x_i < x_j} (2c - a - b + a + b)\Big) \\
&= f_T\Big(\sum_{i,j \in T, x_i = x_j = 0} a + \sum_{i,j \in T, x_i = x_j = 1} b + \sum_{i,j \in T, x_i \neq x_j} c\Big) \\
&= \Psi_T(\mathbf{x}_T).
\end{aligned}$$

In the derivation, we used that each pair (k, ℓ) is counted twice, once as $i = k, j = \ell$ and once as $i = \ell, j = k$.

The P^n functions are still submodular node potentials [9]. For this submodularity, it is crucial that the pairwise potentials ϕ_T (corresponding to edge weights in the CoopCut graph) are *uniform* across the clique. A counterexample for non-uniform weights follows from the graph in Figure 3, where $T = \{x_1, x_2, x_3, x_4\}$: Let all edges that are missing for the clique have weight zero or a small positive weight. This graph corresponds to non-uniform pairwise potentials $\phi_T(x_i, x_j) = w_{(v_i, v_j)}/2$. The resulting clique potential $\Psi_T(\mathbf{x})$ is not submodular, as the caption shows. If f_T is linear increasing, then Ψ_T corresponds to a standard graph cut potential with modular edge weights, and varying weights are allowed.

5.2 Robust P^n potentials

Let $N(\mathbf{x}_T)$ be the number of "deviating" labels in a clique T, i.e., the number of nodes taking the minority label, and let $q \leq |T|/2$. Then the robust P^n potential [36] is defined as

$$\Psi_c(\mathbf{x}_c) = \begin{cases} N(\mathbf{x}_T)a/q & \text{if } N(\mathbf{x}_T) \leq q \\ a & \text{otherwise,} \end{cases}$$

We model this potential via cooperating terminal edges. Let S_1 be the group of all edges (s, v) for $v \in T$, and S_2 the group of all edges (v, t), and

$$f_{S_i}(C) = \min\{|C \cap S_i|,\ q\}a/q$$

for $i = 1, 2$. This is a thresholded discount function for edge weights a/q, $\theta_S = a$ and $g(x) = 0$. For the overall

potential, we set

$$\begin{aligned} f(\delta(\mathbf{x}_T)) &= f_{S_1}(\delta(\mathbf{x}_T)) + f_{S_2}(\delta(\mathbf{x}_T)) - a \\ &= \min\big\{\,|\{i\,|\,x_i = 1\}|,\ q\big\}a/q + \min\big\{\,|\{i\,|\,x_i = 0\}|,\ q\big\}a/q - a \\ &= a + \min\{N(\mathbf{x}_T),\ q\}a/q - a \\ &= \Psi_T(\mathbf{x}_T). \end{aligned}$$

5.3 Related ideas

In Machine learning, *sparsity*, that is, having few nonzero variables, has traditionally been enforced by penalizing the ℓ_1 norm of the vector of variables – the ℓ_0 norm is difficult to optimize in the continuous setting. More recently, *structured sparsity*, where sparsity is with respect to groups of variables, was introduced. There, the ℓ_1 norm is takes over groups to enforce only few groups to be nonzero, and within groups, the ℓ_2 norm is used to distribute the weight throughout the group. These norms are called, depending on the setting, group norms or mixed norms.

In discrete optimization, there is no real analog to these regularization approaches (yet). Nevertheless, counting the number of selected variables (edges) may be seen as a sparsity-enforcing penalty. In that view, our group-based cost is similar to a structured regularization, but in the discrete setting. We will remain at this level of analogy here and not draw any statistical cocnlusions for cuts. Still, cooperative costs can intruduce a preference for a certain structure into the discrete sparsity pattern.

6 Conclusion

We defined a new graph-cut function with *submodular edge weights* that leads to a class of global potentials. Such potentials can express a discrete structured regularization, for example extend the notion of smoothness to homogeneous boundaries. Even though finding a minimum cut for submodular edge costs is NP hard, we propose an efficient approximation algorithm that works well in practice. Furthermore, the experiments show that homogeneity-based smoothness improves the segmentation of objects with complicated boundaries around fine segments, and of partially shaded objects even in grayscale images. Cooperative smoothness can be integrated into many common potentials and combined with constraints. Finally, homogeneity-based regularization is only one example of the richness of cooperative cuts – they have many more potential applications in Computer Vision.

Acknowledgments

We would like to thank Sebastian Nowozin for discussions.

References

[1] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*, 2001.

[2] 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(9):1124–1137, 2004.

[3] C. Rother, V. Kolmogorov, and A. Blake. Grabcut – interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[4] Y. Boykov and G. Funka-Lea. Graph cuts and efficient N-D image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[5] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. In *CVPR*, 2008.

[6] O. J. Woodford, C. Rother, and V. Kolmogorov. A global perspective on map inference for low-level vision. In *ICCV*, 2009.

[7] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[8] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *ICCV*, 2009.

[9] P. Kohli, M.P. Kumar, and P.H.S. Torr. P^3 & beyond: Move making algorithms for solving higher order functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1645–1656, 2009.

[10] A. Schrijver. *Combinatorial Optimization*. Springer, 2004.

[11] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press, 2007.

[12] S. Nowozin and C. H. Lampert. Global connectivity potentials for random field models. In *CVPR*, 2009.

[13] S. Fujishige. *Submodular Functions and Optimization*. Number 58 in Annals of Discrete Mathematics. Elsevier Science, 2nd edition, 2005.

[14] L. Lovász. *Mathematical programming – The State of the Art*, chapter Submodular Functions and Convexity, pages 235–257. Springer, 1983.

[15] H. Narayanan. *Submodular Functions and Electrical Networks*. Elsevier Science, 1997.

[16] A. Krause and C. Guestrin. ICML tutorial: Beyond convexity: Submodularity in machine learning, 2008.

[17] A. Krause and C. Guestrin. IJCAI tutorial: Intelligent information gathering and submodular function optimization, 2009.

[18] A. Krause, P. Ravikumar, and J. Bilmes. NIPS workshop on discrete optimization in machine learning: Submodularity, sparsity and polyhedra, 2009.

[19] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows*. Prentice Hall, 1993.

[20] E.B. Sudderth, M.J. Wainwright, and A.S. Willsky. Loop series and Bethe variational bounds in attractive graphical models. *Advances in neural information processing systems*, 20:1425–1432, 2008.

[21] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE transactions on pattern analysis and machine intelligence*, 29(7):1274–1279, 2007.

[22] D. Freedman and P. Drineas. Energy minimization via graph cuts: Settling what is possible. In *CVPR*, 2005.

[23] S. Fujishige and S. B. Patkar. Realization of set functions as cut functions of graphs and hypergraphs. *Discrete Mathematics*, 226:199–210, 2001.

[24] S. Živný, D. A. Cohen, and P. G. Jeavons. Classes of submodular constraints expressible by graph cuts. In *CP*, 2008.

[25] S. Živný, D. A. Cohen, and P. G. Jeavons. The expressive power of binary submodular functions. *Discrete Applied Mathematics*, 157(15):3347–3358, 2009.

[26] D. M. Greig, B. T. Porteous, and A. H. Seheult. Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society*, 51(2), 1989.

[27] L. S. Shapley. Cores of convex games. *International Journal of Game Theory*, 1(1):11–26, 1971.

[28] S. Jegelka and J. Bilmes. Cooperative cuts: graph cuts with submodular edge weights. Technical Report TR-189, Max Planck Institute for Biological Cybernetics, 2010.

[29] V. Vineet and P J Narayanan. CUDA cuts: Fast graph cuts on the GPU. CVPR Workshop on Visual Computer Vision on GPUs, 2008.

[30] R. Wu, B. Zhang, and M. Hsu. GPU-accelerated large scale analytics. Technical report, HP Laboratories, 2009.

[31] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular functions - i. *Math. Program.*, 14:265–294, 1978.

[32] X. Ren and A. Malik. Learning a classification model for segmentation. In *ICCV*, 2003.

[33] F. Bunke, H. W. Hamacher, F. Maffioli, and A. Schwahn. Minimum cut bases in undirected networks. *Discrete Applied Mathematics*, In Press, 2009.

[34] R. E. Gomory and T. Hu. Multi-terminal network flows. *Journal of the SIAM*, 9(4), 1961.

[35] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *ECCV*, 2004.

[36] P. Kohli, L. Ladickỳ, and P.H.S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.