Joint Optimization of Segmentation and Color Clustering

Ekaterina Lobacheva [1] elobacheva@hse.ru
Olga Veksler [2] olga@csd.uwo.ca
Yuri Boykov [2] yuri@csd.uwo.ca

[1] National Research University
Higher School of Economics,Russia

[2] Computer Vision Group
University of Western Ontario, Canada

Abstract

Binary energy optimization is a popular approach for segmenting an image into foreground/background regions. To model region appearance, color, a relatively high dimensional feature, should be handled effectively. A full color histogram is usually too sparse to be reliable. One approach is to reduce dimensionality by color space clustering. Another popular approach is to fit GMMs for soft color space clustering. These approaches work well when the foreground/background are sufficiently distinct. In cases of more subtle difference in appearance, both approaches may reduce or even eliminate foreground/background distinction. This happens because either color clustering is performed completely independently from segmentation, as a preprocessing step (in clustering), or independently for the foreground and independently for the background (in GMM). We propose to make clustering an integral part of segmentation, by including a new clustering term in the energy. Our energy favors clusterings that make foreground/background appearance more distinct. Exact optimization is not feasible, therefore we develop an approximate algorithm. We show the advantage of including the color clustering term into the energy function on camouflage images, as well as standard segmentation datasets.

1. Introduction

Optimization of binary energy is a common approach for segmenting an image into foreground/background regions [3, 14, 2, 18, 11, 10, 15, 17]. Most segmentation functions include an appearance term to model foreground/background, and a smoothness term to encourage smooth boundaries. Here we focus on appearance models.

We are interested in the case when appearance models are not known in advance, but rather must be estimated from the image to be segmented, like in the GrabCut framework [14, 2]. The appearance parameters have to be estimated together with segmentation, making the problem NP-hard [2], in contrast to the case of known appearance [3],

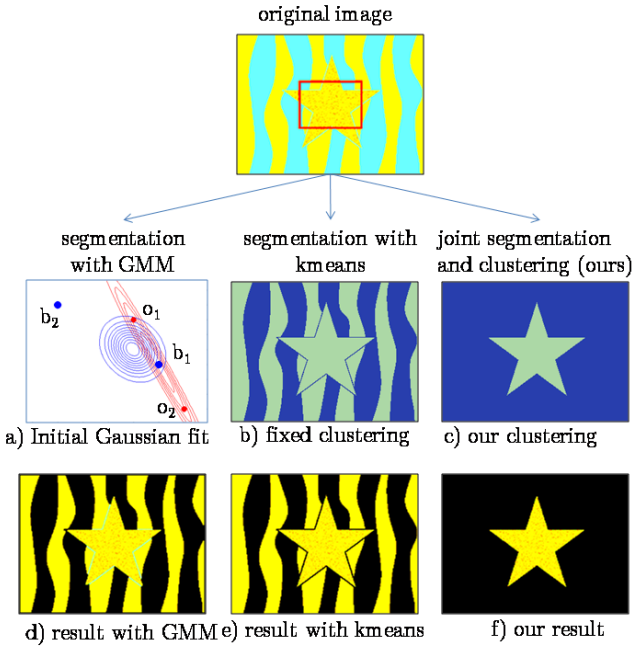


Figure 1. Top: original image with the initialization box in red. The star object has two distinct colors: o_1 and o_2, and the stripy background has distinct colors b_1 and b_2. Colors o_1 and b_1 (in orange spectrum) are similar. Bottom: (a) initial GMM fit with one component for the foreground and background. Blue and red dots show background and foreground colors, respectively. Background color b_1 fits the foreground model better, leading to a segmentation failure in (d); (c) kmeans clusters colors into $\{o_1, o_2, b_1\}$ and $\{b_2\}$. This makes o_1 and b_1 indistinguishable, causing a segmentation failure in (e). Right: we find an adaptive clustering into two components that keeps initial foreground colors (inside red box) and background colors (outside red box) in separate clusters.

To estimate appearance models, one initializes with a rough segmentation, usually a box. The foreground is estimated from the inside and the background from the outside

box. Using the estimated models, the image is segmented with [3], which for fixed appearance models gives an optimal segmentation. Model estimation and segmentation are repeated until convergence to a local minimum [2].

Color is a popular feature for segmentation. There are various approaches for handling color, a relatively high dimensional feature. The simplest approach is to compute a full histogram (normalized), like it is done for gray scale images in [4]. Despite high dimensionality, this can be implemented efficiently with a hash table. However, full color histogram is likely to be sparse and unreliable.

To obtain a smaller size histogram with more reliable bin counts, one can cluster the color space into clusters (or bins) [18]. When the color space is partitioned evenly, this is usually referred to as quantization and the resulting cells are referred to as bins. When the color space is partitioned irregularly, this is usually called clustering and the resulting "cells" are called clusters. In this paper, we use the terms quantization and clustering interchangeably. An advantage of this approach is that the histogram of optimal fit to a given segment can be computed exactly [18] in linear time, making this approach particularly efficient.

Color space partitioning is performed as a pre-processing step, completely independently from the segmentation process. Clearly, some color clusterings enhance the object/foreground difference, while other clusterings reduce it. Consider a simple illustration in Fig. 1. The top shows an image with the initialization box in red[1]. The star foreground object has two distinct orange colors, o_1 and o_2. The striped background region has distinct colors b_1, which is close to o_1, and b_2, which is far from all the other colors.

Suppose that we choose to cluster color space into two clusters. Kmeans (Fig. 1(b)) produces clusters $\{o_1, o_2, b_1\}$ and $\{b_2\}$, since b_2 is far away from all the other colors. This clustering makes the foreground star region completely indistinguishable from the background stripes of color b_1, leading to a segmentation failure shown in Fig. 1(e). A clustering that maintains a good foreground/background separation of the *initial* segmentation (i.e. the box) does exist, namely $\{o_1, o_2\}$ and $\{b_1, b_2\}$. However, this clustering is somewhat worse according to the kmeans objective function, and is impossible to find without considering the clustering objective function together with image segmentation.

Another popular approach that copes with high-dimensionality of color is to use mixtures of Gaussian models (GMM), first proposed in [14]. Even though it is intractable to find an optimal GMM model that best fits a given segmentation, GMM modeling works quite well if the foreground/background appearance are sufficiently distinct. However, it has problems for more subtle cases, sometimes referred to as "camouflage". Consider Fig. 1 again. We use GMM with one component for foreground and one for the background[2]. The initial GMM fit is in Fig. 1(a). The red Gaussian is for the foreground and the blue one for the background. The background color b_1 has a higher probability under the foreground GMM, and therefore at the first iteration, all pixels of color b_1 are assigned to the foreground. This leads to a segmentation failure, see Fig. 1(d). The problem is that at the model refitting stage, the foreground and background are treated independently from each other. The foreground GMM is a good model for a large part of the background, because the background was never considered when fitting the foreground.

To address the disconnect between the clustering and segmentation steps, we propose to include a color clustering term into the energy function, making it an integral part of the segmentation process. Thus our approach jointly optimizes over color clustering, foreground/background appearance models, and the segmentation boundary.

Our energy is designed to encourage a color clustering that makes the foreground and background regions more distinct from each other. Consider Fig. 1 again. We start with the same clustering into two components as kmeans, see Fig. 1(b). However, under clustering in (b), the initial foreground/background segmentation, given by the red box, is not likely (i.e. data term for the background pixels is high). We search for a new clustering that makes the initial segmentation more likely. The result of this 'adaptive' clustering is in Fig. 1(c). Under the appearance resulting from Fig. 1(c), the initial red box segmentation is more likely. When we segment the image, we get the result in Fig. 1(f).

Our intuition is derived from [18, 17], where they provide an interesting interpretation of the energy function that is used for jointly optimizing the appearance and segmentation terms. However, the approaches in [18, 17] assume that the color space has been quantized beforehand. Furthermore, the goals in these papers are different from ours. The main goal in [18] is to develop a dual decomposition approach for joint optimization of the appearance and segmentation that is able to find a global optimum in some instances. The goal in [17] is a tractable (not NP-hard) appearance based algorithm. To achieve it they replace the non-tractable volume-balancing term with application-dependent unary (tractable) terms.

We base our color clustering term on the kmeans objective function [8]. Our aim is to find a clustering that leads to histograms which better separate the foreground region from the background. This is, of course, the chicken and egg problem, since segmentation of the foreground is our goal in the first place. We design a stepwise approach where

[1] Unlike the original framework in [14], in this example we do not hard-constrain the outside of the box to be the background. This assumes that the user roughly centers the box around the object of interest, not necessarily ensuring that all the foreground fits inside the box.

[2] Technically, one component GMM is just a Gaussian. We use a simple image for illustration. However, all arguments transfer to the general case.

we first optimize the appearance together with clustering, and then appearance with segmentation, in a way that guarantees not to increase the energy. For the first step we develop a new optimization scheme that combines the standard kmeans algorithm with the swap moves from [5]. We call this novel algorithm *swap-kmeans*. The second step is the same problem as in [2], and is solved in the same way.

We validate our approach on camouflage images, as well as standard segmentation datasets with ground truth.

2. Energy Function

We now formulate and motivate our energy function.

2.1. Appearance and Color Clustering

We first describe the appearance and color clustering terms and discuss how they interact. Our new clustering term is based on the kmeans objective function and we add it to the standard segmentation energy [3, 2]. The appearance model is based on normalized histograms, like in [3, 18].

Let p be an image pixel and $\mathcal{P}$ the set of all image pixels. Let c be a color occurring in the image, and $\mathcal{C}$ the set of all distinct image colors[3]. Let c_p be the color of pixel p.

We seek to partition the set of all colors $\mathcal{C}$ into k clusters. Let clusters be indexed with integers $\{1, 2, ..., k\}$. For each $c \in \mathcal{C}$, a clustering variable $y_c \in \{1, 2, ..., k\}$ denotes the cluster of color c. We will refer to vector y as the *clustering*.

Given a color clustering y, let $P_i(y)$ be the set of pixels whose colors are assigned to cluster i. That is

$$P_i(y) = \{p \in \mathcal{P} | y_{c_p} = i\}.$$

We compute the center of each cluster i as:

$$\mu_i(y) = \frac{1}{|P_i(y)|} \sum_{p \in P_i(y)} c_p,$$

where $|\cdot|$ stands for the size of a set.

Our kmeans-based clustering energy term is:

$$E_c(y) = \sum_{p \in \mathcal{P}} ||c_p - \mu_{y_{c_p}}||^2. \quad (1)$$

This term encourages similar colors to be grouped together, and it depends only on clustering y.

We now turn to the appearance term. Let $x_p \in \{0, 1\}$ be the segmentation label of pixel p where 0 denotes the background and 1 the foreground. Let θ^0 and θ^1 be the appearance models for the background and foreground, respectively. Our appearance models are based on color clustering. That is for pixel p, with its cluster index y_{c_p}, $Pr(y_{c_p}|\theta^0)$ is the probability that p is the background pixel and $Pr(y_{c_p}|\theta^1)$ is the probability that p is a foreground pixel. The appearance term is:

$$E_a(x, y, \theta^0, \theta^1) = \sum_{p \in \mathcal{P}} -logPr(y_{c_p}|\theta^{x_p}), \quad (2)$$

and it depends on clustering y, segmentation x, and appearance models θ^0, θ^1. This is different from formulation in [18], where appearance does not depend on clustering.

As in [18], rewrite Eq. (2) so that it depends only on histogram counts. For fixed segmentation x and clustering y, the optimal distributions θ^0, θ^1 that minimize Eq. (2) are normalized histograms computed separately over foreground and background regions. Thus computing optimal models θ^0, θ^1 is trivial and efficient, unlike fitting GMMs.

For $s \in \{0, 1\}$, and color cluster $i \in \{1, 2, \ldots, k\}$ let

$$n_i^s(x, y) = |\{p \mid x_p = s \text{ and } y_{c_p} = i\}|.$$

That is $n_i^s(x, y)$ is the number of pixels that have color in cluster i and segmentation label s, under fixed y and x. Let

$$n^s(x) = |\{p \mid x_p = s\}|,$$

that is $n^0(x)$ and $n^1(x)$ are the number of pixels in the foreground and background, respectively, under segmentation x. Then the optimal θ^0, θ^1 that minimize Eq. (2) are:

$$Pr(y_p|\theta^s) = \frac{n_i^s(x, y)}{n^s(x)}. \quad (3)$$

Substituting Eq. (3) into Eq. (2), and summing over color clusters instead of pixels, the appearance term is rewritten:

$$E_a(x, y) = \min_{\theta^0, \theta^1} E_a(x, y, \theta^0, \theta^1) \quad (4)$$

$$= \sum_{i \in \{1,..,k\}} \sum_{s \in \{0,1\}} -n_i^s(x, y) \log \frac{n_i^s(x, y)}{n^s(x)} \quad (5)$$

Regrouping the terms in Eq. 5, we get:

$$E_a(x, y) = \sum_s h(n^s(x)) + \sum_i \sum_s -h(n_i^s(x, y)), \quad (6)$$

where $h(a) = a \log(a)$. The first sum in Eq. 6 is convex and maximized when half of the pixels in x are labeled as foreground, creating a bias towards a balanced segmentation. This sum is independent of the clustering variable y.

For a fixed cluster i, the inner part of the second sum, i.e $-h(n_i^0(x, y)) - h(n_i^1(x, y))$, is concave. It is minimized when all pixels in cluster i are assigned to either foreground or background. Thus the second sum is minimized when foreground and background pixels are in separate color clusters. The second sum depends on x and y.

We are now ready to discuss our difference from [18], and from the previous histogram based appearance approaches. In [18], the appearance term in Eq. (6) does

[3] $\mathcal{C}$ can be computed efficiently using hash tables.

Figure 2. Left: original image with initialization box in red. Middle: result with a smaller λ_c, segmentation gets stuck immediately close to the initialization box. Right: result with a larger λ_c, colors similar to those in the initial box prefer to group together.

Figure 3. The input image with five initializations boxes shown on the left. Other images are results corresponding to these initializations, color coded. The red box result is of lowest energy.

not depend on clustering, since they assume clustering is fixed at the preprocessing step. We make Eq. (6) depend on clustering y, and therefore we can search over clusterings y that make the term in Eq. (6) smaller. This corresponds to *adaptively* choosing a clustering that makes foreground and background appearance more distinct, preventing clusterings that collapse foreground/background distinctions.

Consider the example in Fig. 1 again. Let x^* be the ideal segmentation, that is $x_p^* = 1$ for all pixels p in the star, and $x_p^* = 0$ for all other pixels. Let y_1, y_2 be the color clusterings in Fig. 1(b) and Fig. 1(c), respectively. Then $E_c(y_1) < E_c(y_2)$ since y_1 is a better color clustering. However, $E_a(x^*, y_1) > E_a(x^*, y_2)$, since under x^* and y_2, the normalized histogram counts are as sharply peaked as possible for segmentation x^*. Let

$$E_{ac}(x, y) = \lambda_c E_c(y) + E_a(x, y),$$

where λ_c balances the relative importance of E_a and E_c. For a certain range of λ_c, $E_{ac}(x^*, y_2)$ has a smaller energy than $E_{ac}(x', y_1)$ for any other segmentation x'. This discussion can be repeated when x^* is not too far from the ideal segmentation, for example, for the initialization box in Fig. 1. Thus a clustering that favors good background/foreground separation may be preferred.

For a fixed segmentation x, if y is allowed to vary, we can search for a clustering y such that each cluster either mostly contains the colors assigned to the background pixels, or mostly the colors assigned to the foreground pixels. This would make little sense if we did not jointly optimize E_a with E_c. Indeed, without the clustering term, for any initial segmentation x', we would assign all colors c that are completely (or mostly) in the foreground to the same single cluster 1 and all the colors c that are completely (or mostly) in the background to another cluster 2, creating a perfect (or almost perfect) color separation and thus making appearance energy $E(x', y)$ very low. We would be likely immediately stuck at x'. However, E_c discourages assignment of different colors to the same cluster. Parameter λ_c controls how much we prefer to keep different colors separate. The effect of λ_c is illustrated in Fig. 2.

There is also a balancing of the number of clusters k between E_a and E_c. E_a prefers less clusters, since this lowers its energy. E_c, on the other hand, prefers a larger k, since with more clusters, its energy is lower.

2.2. Complete Energy

All that is missing is the standard [4, 14] pairwise term that encourages segmentation x to be spatially coherent:

$$E_s(x) = \sum_{(p,q)\in\mathcal{N}} w_{pq}|x_p - x_q|, \tag{7}$$

where $\mathcal{N}$ is a neighborhood system consisting of ordered pairs of pixels, and w_{pq} is defined as in [4, 14]:

$$w_{pq} = \exp^{-\frac{||C(p)-C(q)||^2}{2\sigma^2}},$$

where σ is computed as the average of $||C(p)-C(q)||^2$ over all neighboring pixel pairs in the input image.

The complete energy is:

$$E(x, y) = \lambda_s E_s(x) + E_a(x, y) + \lambda_c E_c(y), \tag{8}$$

where λ_s and λ_c weight relative importance of each term.

The energy in [18] is as in Eq. 8 but without the clustering term and without optimization over clusterings y. Recall that the appearance term has a balance towards equally partitioned background and foreground. The addition of the clustering term does not effect this bias since $E_c(y)$ does not depend on x. Such energies can be used for background/foreground separation without user interaction, since an empty segmentation x carries a high cost. We discuss optimization without user interaction in Sec. 3, and in Sec. 4 we describe what changes if there is user guidance.

3. Optimization

The energy in Eq. 8 is hard to optimize, even if we drop the last term, or if we drop the first two terms. We develop a stepwise optimization procedure. In the first step, we optimize over clusterings y keeping segmentation x fixed. In the second step, we optimize over segmentations x, keeping clustering y fixed. Convergence of this iterative procedure is guaranteed, since each step reduces the energy.

3.1. Initialization

We start with x initialized to the foreground in some rectangular box and background outside the box, see Fig. 3. We try five different initialization boxes, placed as illustrated in Fig. 3, and run the optimization procedure described in this section for each initialization. Initialization boxes are placed in the central portion of the image. This reflects the prior knowledge that foreground objects are often centrally located. We choose the lowest energy segmentation. Two iterative steps are described in Sec. 3.2 and 3.3.

3.2. Optimizing over y

Fixing segmentation x, to optimize Eq. 8 over y, we need to consider only $E_{ac}(x,y) = E_a(x,y) + \lambda_c E_c(y)$, since $E_s(x)$ does not depend on y. If only the clustering term E_c involved variables y, then we could apply the standard kmeans algorithm to optimize E_c, which finds a local minimum very effectively. However, kmeans algorithm is not directly applicable because changing y also changes $E_a(x,y)$. We develop an optimization procedure which is a combination of kmeans and a popular swap move algorithm [5].

Let us first review the kmeans algorithm. Starting with some initial cluster centers μ_i, for each color c, we assign y_c to cluster i with the closest μ_i. That is, $y_c = \text{arg}min_i ||c - \mu_i||^2$. This is called the assignment step. Next each μ_i is updated with the mean of all colors currently assigned to it:

$$\mu_i = \frac{1}{P_i(y)} \sum_{p \in P_i(y)} c_p.$$

These steps are iterated until convergence.

Swap-kmeans We optimize E_{ac} in a kmeans fashion. There are two steps, updating of the means and cluster assignment. Updating the means is just as in kmeans, since this step does not change y, and therefore, it does not change $E_a(x,y)$. Furthermore, updating the means is guaranteed not to increase $E_c(y)$, and, therefore, also energy $E_{ac}(x,y)$.

The next step is cluster assignment. We adapt the swap algorithm from [5] for this step. Let us review the swap algorithm first. Given a multi-label energy defined over some set of sites, the swap algorithm cycles through all pairs of labels α, β and finds the optimal subset of sites currently labeled as α to switch to β together with the optimal set of sites currently labeled as β to switch to α. That is sites that are currently labeled as α or β are allowed to exchange their labels in an optimal manner. The swap move is performed efficiently with graph cuts [5] .

In our case, the sites are the image colors $c \in \mathcal{C}$, and the labels are color cluster indexes $\{1, 2, ..., k\}$. Thus we iterate over all cluster index pairs (i,j) and try to find the best possible re-assignment of cluster indexes i, j among all the colors c that currently have $y_c = i$ or $y_c = j$.

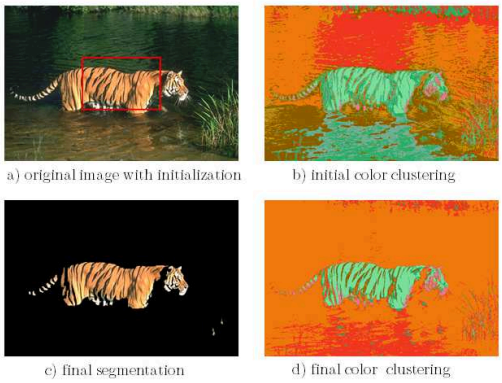


Figure 4. Color clustering illustration. Initial and final color clusterings in (b) and (d) are visualized with random colors for clarity. Notice the initial clustering in (b) puts many tiger colors together with its reflection colors in one cluster, visualized with greenish color. The final clustering after optimization described in Sec. 3.2 is in (d). Notice that now these colors are well separated.

Recall the definition of $E_a(x,y)$ in Eq. 6. The only terms in $E_a(x,y)$ that are effected by the (i,j)-color swap are:

$$-h(n_i^0) - h(n_j^0) - h(n_i^1) - h(n_j^1), \tag{9}$$

For brevity, we drop the dependence on x, y from n_i^s's.

Assume that all pixels that have equal color are assigned either to the foreground or the background under current segmentation x, or, at least, a large majority of pixels with the same color have the same segmentation label. This is usually satisfied in practice during the execution of the algorithm, since the same color is unlikely to be evenly split between foreground and background under any appearance model. The exception is the initialization step, but such initializations are easy to avoid in practice. Under this assumption, in the (i,j)-color swap, the first two terms in Eq. 9 are independent of the last two terms and therefore the first two and the last two can be optimized separately.

Now recall the definition of $E_c(x,y)$ in Eq. 1. In $E_c(y)$, the (i,j)-color swap effects only the terms:

$$\sum_{p \in P_i(y)} ||c_p - \mu_i||^2 + \sum_{p \in P_j(y)} ||c_p - \mu_j||^2. \tag{10}$$

We rewrite Eq. 10 in a more convenient form, but first more notation. Let m_c be the number of pixels that have color c and let $C_i(y)$ be the set of colors in cluster i under clustering y. For brevity we write C_i. Eq. 10 can be written:

$$\sum_{c \in C_i} m_c ||c - \mu_i||^2 + \sum_{c \in C_j} m_c ||c - \mu_j||^2. \tag{11}$$

Let us describe optimization of just the first two terms in Eq. 9, since optimization of the last two terms is identical.

We call these two cases the background (i, j) swap and the foreground (i, j) swap, respectively. Let C_i^0 be the colors that are currently assigned to cluster i and pixels with these colors belong to the background, or at least the majority of them belongs to the background under current segmentation x (recall our assumption above). Then only the colors in $C^m = C_i^0 \cup C_j^0$ participate in the move. Let v be the size of C^m. Let us introduce binary variables $z_1, z_2, ..., z_v$, one z_l for each $c \in C^m$. Let $g(l)$ be the mapping from variable z_l to the actual color it corresponds to in C^m.

The interpretation of these binary variables is as follows. If $z_l = 0$, this means that the corresponding color $g(l)$ is assigned to cluster i . If $z_l = 1$, then color $g(l)$ is assigned to cluster j. Let m_c^0 be the number of the background pixels that have color c. We define the binary move energy:

$$E(z) = \sum_{l \in \{1,...,v\}} m_{g_l} ||g(l) - \mu_{z_l}||^2 - m_{g_l}^0 h(n_0^0) - m_{g_l}^0 h(n_1^0), \tag{12}$$

where n_0^0 is the number of variables z_l assigned to 0 and n_1^0 is the number of z_l assigned to 1[4].

The first term in Eq. 12 is a unary term, and the last two terms are concave cardinality functions of the number of variables z_l assigned to label 1. With an addition of auxiliary variables, such functions can be optimized with an arbitrary degree of accuracy with graph cuts [9]. Since we do not have pairwise cliques, optimization does not require graph cuts and is very efficient. We enumerate all possible assignment to the auxiliary variables, and for each such assignment, the energy in Eq. 12 is optimized by choosing the best label for each z_l individually. Finally we choose the assignment corresponding to the lowest energy out of all enumerated auxiliary pixel assignment cases. We found it sufficient to have four auxiliary variables, giving us only a rough approximation to the concave term. However, this makes enumeration more efficient than graph cuts.

Due to the greediness of our approach, after a foreground and background (i, j) swap, it may happen that either cluster i or cluster j is empty. In this case, we simply assign all the colors with the majority of pixels in the foreground to cluster i and all the colors with the majority of pixels in the background to j. This is guaranteed to keep energy $E_{ac}(x, y)$ unchanged, but lowers $E_c(y)$. Optimization over clustering y is illustrated in Fig. 4.

3.3. Optimizing over x

In this step, we find x that minimizes our energy E, keeping y fixed to the values computed at the previous step (Sec. 3.2). Optimizing over x effects E_s and E_a. This step is performed just like in [2], iteratively. First we compute the new x' that optimizes $E_s(x) + E_a(x, y)$ using the graph cut algorithm of [3]. We then update $E_a(x', y)$ in Eq. (5) to reflect the new histogram counts induced by the new x'. This is guaranteed not to increase the total energy. We iterate computing x' and updating histogram counts until a local minimum of $E_s(x) + E_a(x, y)$ is reached.

[4]For brevity, we omit from notation the dependence of n_0^0 and n_1^0 on z.

4. Experimental Results

We evaluate our segmentation energy with the clustering term on camouflage images as well as standard interactive and non-interactive (figure-ground or saliency) segmentation datasets with ground truth. The standard benchmarks do not contain many difficult (camouflage) images, but our energy still shows some improvement over the standard energy, i.e. the one without the color clustering term.

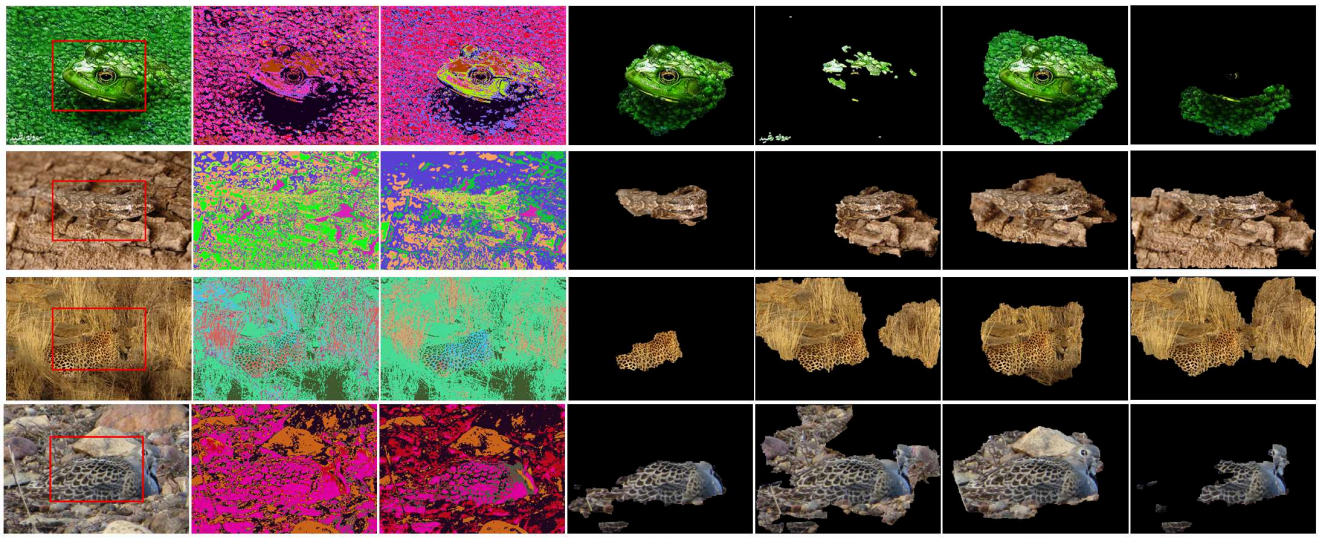
We used two datasets with ground truth ([1], [14]), as well as animal camouflage images. For the datasets with ground truth, for each algorithm, we chose the parameters that optimize the error rate, as is common practice in the field. For the camouflage images, we used the parameters that work best on the database from [14].

4.1. Results on Camouflage Images

In Fig. 5, we show the results of our approach on images with camouflage. In Fig. 5(a) we show the original image with the central box used for initialization. In Fig. 5(b) we show the initial color clustering and in Fig. 5(c) the final color clustering estimated by our algorithm. Notice that the foreground object is much more distinct in (c) compared to the initial clustering in (b). In Fig. 5(d) we show our segmentation results. Our results are much better compared to the fixed (non-optimized) color clusterings, which we show in (e,f,g). They correspond, respectively, to k-means, quantized histogram, and full histogram clusterings.

4.2. Interactive Benchmarks

In this section, we use the GrabCut [14, 13] dataset for evaluation. We place initialization boxes for x only inside the area indicated as the foreground or uncertain by the user. The areas indicated as the background (if any), are hard-constrained. Note that our background is initialized from all hard-constrained background pixels, which is somewhat different from other papers. In table 1, we compare our approach ("Ours") with the energy optimization without the clustering term. Our energy contains three terms, the standard E_s and E_a plus the clustering term E_c that we introduced into the segmentation energy. We compare to optimization of the same energy but without the use of clustering term E_c, under different appearance models. Namely, we compare to GMMs("GMM"); full color histograms ("Full Hist."); color quantization in equal bins ("Color Quant."); and, finally, color clustering with kmeans ("Color Clust."). For all energies, E_s term is exactly the same, while the E_a term is computed according the appearance model being used. To make comparisons fair, for each



(a) (b) (c) (d) (e) (f) (g)

Figure 5. Results on camouflage images: (a) original image with the central box; (b) initial color clustering; (c) final color clustering of our algorithm; (d) our results; (e) results with k-means clustering; (f) results with quantized histogram; (g) results with full histogram

algorithm, we found parameters that minimize the pixel error computed using ground truth.

Out of the histogram-based methods, full histogram is the worst, it is likely to have unreliable bin counts. Color clustering is slightly better than color quantization. But our method, with all three energy terms, performs the best. Our running time is nearly as efficient as the other histogram based methods because we are able to implement all the optimization related to the $E_c + E_a$ terms very efficiently based on histogram count manipulations, which is proportional to the number of distinct colors in the image, usually much smaller than the number of pixels.

Our method does not achieve the best error rates [17] on the GrabCut dataset[5]. However, our motivation here is to show the advantages of our energy vs. non-optimized color clustering models, and not (necessarily) to achieve the state-of-the art results on this particular dataset.

All methods will have a variation in performance relative to parameter settings. Our approach is relatively stable. Varying λ_s from 200 to 600 in steps of 100, and λ_c from 0.005 to 0.05 in steps of 0.005, gives error rates in the range from 7.5% to 10.2%.

4.3. Non-Interactive Benchmarks

We now evaluate our approach in a non-interactive manner, i.e. for salient object segmentation. We use the saliency datasets from [1], original images from [12]. In addition,

[5] Also note that [17] excludes one of the images and has a heuristic that adapts a volume balancing weight parameter to each image. While we could also develop adaptive heuristics, we find that comparison on the same energy but with and without a clustering term is a cleaner approach.

	% pixel err	aver. time (sec)
$E_s + E_a + E_c$ (Ours)	7.5	10.4
$E_s + E_a$: (GMM)	8.9	5.4
$E_s + E_a$: (Full Hist.)	12.3	3.6
$E_s + E_a$: (Color Quant.)	8.5	4.8
$E_s + E_a$: (Color Clust.)	8.2	7.2

Table 1. Pixel error percentages and average running times on the GrabCut database, in interactive setting.

	GrabCut	Achanta
$E_s + E_a + E_c$ (Ours)	8.0	6.0
$E_s + E_a$: (GMM)	9.4	7.5
$E_s + E_a$: (Full Hist.)	13.8	10.5
$E_s + E_a$: (Color Quant.)	9.3	8.2
$E_s + E_a$: (Color Clust.)	8.8	7.6

Table 2. Pixel error percentages. Notice that the GrabCut database is used here without the user provided bounding box.

we also run experiments without the interactivity information on the GrabCut database, since there are many images with a clear "salient" object. The pixel error counts are in Table 2, and the methods we compare to are the same as in Table 1. As far as we know, we are the first to report non-interactive results on the GrabCut dataset.

Achanta dataset was constructed specifically for saliency and therefore most often contains an obvious salient object. Grabcut is for interactive segmentation and, often also contains an obvious object of interest. Again, the full histogram performs the worst out of $E_s + E_a$ methods. Our method

performs the best, with a small but noticeable gap.

For the Achanta database, a more standard comparison metric published in the literature is the F-measure, computed as $\frac{1.3 \cdot Precision \cdot Recall}{0.3 \cdot Precision + Recall}$. The methods in Table 2 have F-measures, respectively, $87.0, 81.9, 79.7, 81.2, 82.0$ with our approach having again the best F-measure. Notice that this F-measure is very close to the state of the art F-measure of 90 % given by the algorithm [6] specifically designed for saliency detection, whereas our energy is rather generic.

Conclusions and Future Work

We propose a novel energy function that makes color clustering an integral part of the segmentation energy, not a separate pre-processing step. Our energy function with a clustering term favors clusterings that make foreground/background appearance more distinct. Since exact optimization of our energy is not feasible, we develop an approximate step-wise algorithm. As part of our optimization approach, we adapt the popular swap move algorithm to our energy function. We show the advantage of including the color clustering term into the energy function on camouflage images, as well as standard segmentation datasets.

Currently, we use a greedy block-coordinate descent approach to optimize NP-hard energy, which can get stuck in a poor local minimum. To some degree, we address the problem by using several initializations, Fig. 3. A more global approach, for example based on LP relaxation, should give more significant improvements. Even individual BCD steps could be improved, for example the step in Sec. 3.3 could be better optimized using the pseudo-bound method [16], and we could develop expansion moves for the step in Sec.3.2, which usually work better than the swap moves.

Another direction is to justify our energy from the the MDL principle. This may give useful insights on parameter settings, as well as useful variations of the energy terms. Note that a special case of our joint segmentation and binning energy (8) can be written using information theoretic concepts:

$$(H(y) + H(c|y) - MI(x,y)) + \lambda_s E_s(x) \qquad (13)$$

where binning entropy and mutual information $H(y) - MI(x,y) = H(y|x) \sim E_a$ relate to our likelihoods term, while conditional entropy $H(c|y)$ corresponds to our K-means color clustering term E_c. Here we assume that each pixel's color c_p is unique so that there is one-to-one correspondence between segmentation variables x_p and clustering variables $y_p := y_{c_p}$. This allows to formally define mutual information $MI(x,y)$ between segmentation x and color binning y. Adding to this energy one more *sparsity* term $||y||_0$ counting the number of distinct color clusters (or bins) we get objective

$$\gamma||y||_0 + (H(y) + H(c|y) - MI(x,y)) + \lambda_s E_s(x) \quad (14)$$

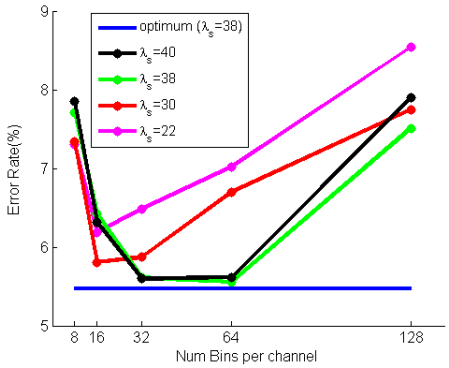


Figure 6. Error rates for pseudo-bound optimization [16] of GrabCut energy with fixed regular bins of different size (horizontal axis). Errors are averaged over GrabCut dataset except the "cross" image often omitted (bounding box covers the whole image). The curves correspond to different values of the smoothness parameter λ_s. The blue line indicates value 5.4 - the average error when optimal bin size is automatically selected for each image based on MI energy (14) for $\lambda_s = 38$. Here we search only over binnings with regular equal bins.

which can be directly derived from MDL principles for a straightforward image compression scheme [7]. One advantage of such energies is that parameters γ and λ_s follow from the scheme. Figure 6 shows preliminary results for an MDL energy as in (14). The energy is used to automatically determine a color bin resolution (number of bins per channel) optimal for each image when computing segmentation with GrabCut energy [14], which is a subset of terms in (14).

Acknowledgments

We are thankful for generous support by Canadian NSERC Discovery and RTI Programs. E. Lobacheva was supported by RFBR project No. 15-31-20596 (mol-a-ved) and by Microsoft: Moscow State University Joint Research Center (RPD 1053945).

References

[1] R. Achanta, F. Estrada, P. Wils, and S. Ssstrunk. Salient region detection and segmentation. In *International Conference on Computer Vision Systems*, pages 66–75, 2008. 6, 7

[2] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *ECCV*, pages Vol I: 428–441, 2004. 1, 2, 3, 6

[3] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *International Journal Computer Vision*, 70(2):109–131, 2006. 1, 2, 3, 6

[4] Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *ICCV*, pages 105–112, 2001. 2, 4

[5] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001. 3, 5

[6] M.-M. Cheng, G.-X. Zhang, N. J. Mitra, X. Huang, and S.-M. Hu. Global contrast based salient region detection. In *IEEE CVPR*, pages 409–416, 2011. 8

[7] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96:1–27, Jan. 2012. 8

[8] J. A. Hartigan and M. A. Wong. A K-means clustering algorithm. *Applied Statistics*, 28:100–108, 1979. 2

[9] P. Kohli, L. Ladicky, and P. H. S. Torr. Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009. 6

[10] V. S. Lempitsky, A. Blake, and C. Rother. Image segmentation by branch-and-mincut. In *ECCV*, pages 15–29, 2008. 1

[11] Y. Li, J. Sun, C. Tang, and H. Shum. Lazy snapping. *ACM Trans. Graph.*, 23(3):309–314, 2004. 1

[12] T. Liu, J. Sun, N.-N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. In *Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 7

[13] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference Computer Vision*, volume 2, pages 416–423, July 2001. 6

[14] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004. 1, 2, 4, 6, 8

[15] C. Rother, T. P. Minka, A. Blake, and V. Kolmogorov. Cosegmentation of image pairs by histogram matching - incorporating a global constraint into mrfs. In *CVPR (1)*, pages 993–1000, 2006. 1

[16] M. Tang, I. B. Ayed, and Y. Boykov. Pseudo-bound optimization for binary energies. In *ECCV*, pages 691–707, 2014. 8

[17] M. Tang, L. Gorelick, O. Veksler, and Y. Boykov. Grabcut in one cut. In *ICCV*, pages 1769–1776, 2013. 1, 2, 7

[18] S. Vicente, V. Kolmogorov, and C. Rother. Joint optimization of segmentation and appearance models. In *International Conference on Computer Vision*, 2009. 1, 2, 3, 4