Recursive MDL via Graph Cuts: Application to Segmentation

Lena Gorelick Andrew Delong Olga Veksler Yuri Boykov

Department of Computer Science University of Western Ontario, Canada

Abstract

We propose a novel patch-based image representation that is useful because it (1) inherently detects regions with repetitive structure at multiple scales and (2) yields a parameterless hierarchical segmentation. We describe an image by breaking it into coherent regions where each region is well-described (easily reconstructed) by repeatedly instantiating a patch using a set of simple transformations. In other words, a good segment is one that has sufficient repetition of some pattern, and a patch is useful if it contains a pattern that is repeated in the image.

Our criterion is naturally expressed by the well-established minimum description length (MDL) principle. MDL prefers spatially coherent regions with consistent appearance and avoids parameter tuning. We minimize the description length (in bits) of the image by encoding it with patches. Because a patch is itself an image, we measure its description length by applying the same idea recursively: encode a patch by breaking it into regions described by yet simpler patches. The resulting hierarchy of inter-dependent patches naturally leads to a hierarchical segmentation.

We minimize description length over our class of image representations (all patch hierarchies / partitions). We formulate this problem as a recursive multi-label energy. Existing optimization techniques are either inapplicable or get stuck in poor local minima. We propose a new hierarchical fusion (HF) algorithm for energies containing a hierarchy of 'label costs'. Our algorithm is a contribution in itself and should be useful for this new and difficult class of energies.

1. Introduction

Automatic image segmentation is classical problem in computer vision. Many segmentation methods maintain an appearance model for each segment by computing a combination of color, texture, and shape statistics. In such approaches, a good segment is one that has a consistent appearance, while a good appearance model is one that describes the segment well [15, 21, 1]. Simultaneously finding a good segmentation and good appearance models is difficult, and local minima are the best we can hope for even with the simplest class of appearance models [15].

In unsupervised segmentation the complexity of the partition must be balanced against the complexity of appearance models; this avoids a trivial segmentation with the image itself as the model. Appearance models are traditionally kept simple by explicitly restricting the number and class of such models (histograms, mixture of gaussians [15, 21]). Partitions are kept simple by penalizing some combination of boundary length, curvature, or segment size.

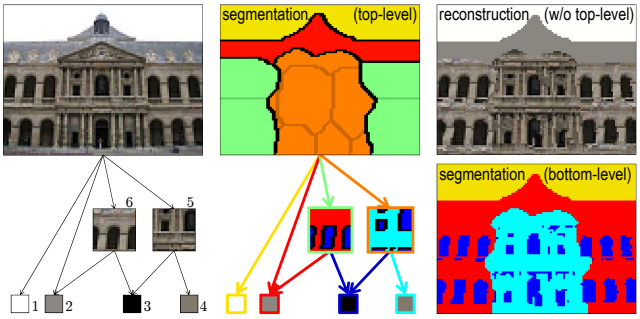

Figure 1. Our representation breaks the image into regions (yellow,red,orange,green) that are well-described by patches (1,2,5,6). Patches 5 & 6 are recursively segmented using 1-pixel patches. We detect repetition and allow arbitrary dependencies among patches. Top-level segmentation: dark boundaries delineate regions represented by different patches; faded boundaries delineate different transformations to match a patch to the image. The bottom-level segmentation is a consequence of the MDL patch hierarchy.

Leclerc [12] was first to argue that minimizing boundary length corresponds to applying the *minimum description length* (MDL) principle to partitions. The MDL principle also prefers segments with consistent appearance because their interior can be described with fewer bits using one appearance model. Subsequent works [23, 6] applied MDL to penalize the number of required appearance models in addition to boundary length. All these works can be interpreted as optimizing over a class of image representations, or *encodings*, that describe the image by a partition and a flat list of appearance models. However, they do *not* apply MDL to the complexity of appearance models themselves.

Recursive Patch-Based Representation We propose a patch-based image representation that can capture a hierarchy of repeated patterns and provides multiple segmentations. Patches preserve spatial information and have been shown useful for many computer vision tasks, *e.g.* super-

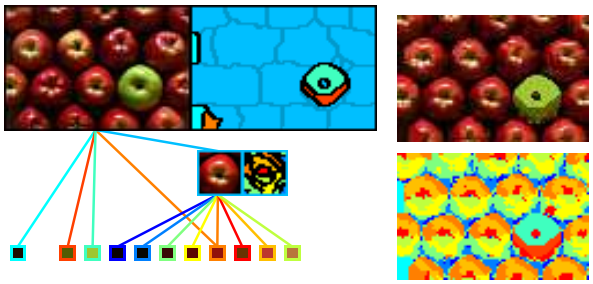

Figure 2. Representation of image containing near-regular pattern. The large blue region is well-described by repeatedly transforming the apple patch to the image; faded boundaries delineate regions that use a constant transformation rule (explained in Sec. 3.2).

resolution and de-noising [8], image re-targeting [2] and completion [18]. One of our main contributions is incorporating patch-based appearance models within an MDL framework. Unlike the aforementioned MDL segmentation methods [12, 23, 6] our patch-based appearance models can be arbitrarily complex so their internal complexity must be explicitly measured and penalized to avoid trivial solutions.

We describe an image by breaking it into coherent segments where each segment is well-described (encoded) by repeatedly instantiating a patch using a set of simple transformations. Because a patch is itself an image, we naturally measure its complexity by applying the same idea recursively, *i.e.* we segment the patch and encode it using even simpler patches. This recursive process results in a set of inter-patch dependencies that form a rooted directed acyclic graph (RDAG[1]) with the input image being the root. Each level in the RDAG hierarchy induces a segmentation (see Fig. 1). Our hierarchy represents dependencies among patches and is *not* a hierarchy of regions in the image. This is novel and fundamentally different from bottom-up merging approaches to hierarchical segmentation, *e.g.* [14].

We adopt the MDL principle and seek a representation requiring the fewest bits to encode the original image. This objective corresponds to minimizing a recursive multi-label energy (Sec. 3). When applied to our class of representations, the MDL criterion:

1. naturally leads to recursion and thereby a hierarchy,
2. accounts for complexity of appearance models in a novel and principled way,
3. avoids parameter tuning common in segmentation, and
4. provides compact description of natural images.

Fig.1 shows an RDAG with only two non-trivial patches, and yet these small patches can describe the highly-structured regions by undergoing only a few transformations. The joint description length (dependencies, partitions, transformations, color) is more compact than the raw image data and thus favored by MDL. Moreover, when the entire image is covered by near-regular texture, one small patch is enough to represent most of the image (Figure 2). Figures 8, 11 show how our recursive MDL formulation can detect more complex repetitive structures.

Our representation captures a hierarchy of color, shape, and texture comprising the image. Information at the bottom of the hierarchy is naturally propagated to higher levels until the original image is fully described. Propagating partition information results in a hierarchy of segmentations. Note that although our patches are square, the partitions can be arbitrarily shaped to follow perceptually meaningful boundaries when mapped to the image.

Furthermore, if we propagate partitions *and* color information, we can reconstruct the image to varying degrees of accuracy as follows: first decode the raw pixels, then decode the simplest patches, working upwards until the root of the RDAG (the original image). If we drop information from our representation, e.g. the color information at the top level, we can still try to reconstruct the image using transformations of the remaining patches (Figs.1,2, top-right).

Hierarchical Fusion Algorithm To compute an image representation of minimal length we must simultaneously optimize over all possible partitions, all possible transformations, and all subsets of possible patches. We show that our minimization problem is related to a class of multi-label energies with "label subset costs" described by Delong *et al.* [6]. The label costs in our energy are defined over very large subsets. To the best of our knowledge, no existing method can optimize our energy effectively, including the variant of α-expansion in [6].

We propose a new *hierarchical fusion* (HF) algorithm to better optimize energies that contain label costs defined on large subsets. This algorithm first optimizes a collection of sub-energies determined by the hierarchy of label costs in the energy. The result is a set of labelings that are then stitched together via α-expansion with label costs. This process (Sec. 4.2) is specifically designed to avoid local minima that are problematic for the method in [6], and has provably better optimality guarantees [5]. Though we introduce HF in the context of segmentation, it should be effective in any setting where labels are naturally grouped into subsets, *e.g.* hierarchical clustering / model-fitting.

2. Additional Related Work

Wang *et al.* [19] build a "condensed epitome" along with a transformation map, but their goal is image compression and real-time decompression for rendering. The work of Jojic *et al.* [10] introduced image epitomes for a number of applications, including figure/ground segmentation. However, epitome-based methods require considerable effort to arrange salient image data into a small 2D chart (in [10] the size is fixed), whereas our representation is an RDAG of inter-dependent patches and requires no such '2D packing'.

Detecting repetitive structures is an important problem

[1] An RDAG is similar to a tree, except nodes can share subtrees.

in its own right [8, 2, 18]. Hays *et al.* [9] proposed a method to discover a lattice of near-regular texture (fabric, fence, window), and Wu *et al.* [20] rely on detecting vanishing points and architecture-specific symmetries. Zeng & Van Gool [22] use point-wise repetition to improve multi-label segmentation. We are first to combine patch-based appearance models with the MDL principle for segmentation.

There are number of hierarchical segmentation methods, e.g. [16, 14]. In these works, a segment's appearance is described by a vector of color/texture statistics determined by the hierarchy of its sub-segments. These methods often find good segmentations, but do not search for repetitive content in the image nor seek compact representations.

On an algorithmic level we are closely related to α-expansion methods [3, 6] and in particular the fusion moves of Lempitsky *et al.* [13] and hierarchical graph cut process of Kumar & Koller [11]. Like [11] we apply fusion moves in a bottom-up hierarchical fashion, but unlike [13, 11] our fusions are guided by the structure of our label costs. Again, the entire motivation for our HF algorithm is to better optimize energies with a hierarchy of "label subset costs," an element entirely missing from [13, 11].

3. Our Recursive MDL Image Representation

Our aim is to find an image representation of minimal description length and to thereby achieve a good segmentation. We begin by proposing a general recursive energy that evaluates the description length of an image w.r.t. a set of patches (Sec. 3.1). We then show more precisely how each recursive step encodes its corresponding patch by breaking into regions described by other patches (Sec. 3.2). This recursive process naturally leads to an RDAG among patches via their partitions.

3.1. MDL Representation via Recursive Energy

We express the description length of an image I w.r.t. a fixed set of patches $\mathcal{S} = \{I^1, I^2, \ldots\}$ *recursively* as

$$\mathring{E}(\mathcal{S}; I) = \min_f E_{\text{enc}}(f; \mathcal{S}, I) + \min_{I' \in \mathcal{S}} \mathring{E}(\mathcal{S} \setminus \{I'\}; I') + \lg|\mathcal{S}| \quad (1)$$

$$\mathring{E}(\{\}; I) = E_{\text{direct}}(I). \quad (2)$$

The first term in (1) minimizes the description length E_{enc} of encoding the image I by using patches from $\mathcal{S}$ via labeling $f : \Omega(I) \rightarrow \mathcal{L}$ that maps each image pixel to a discrete label that identifies a transformed patch. The second term in (1) recursively defines the total description length of the set $\mathcal{S}$ itself. This term is minimized over all possible orderings of repeatedly eliminating one patch from the set and encoding it using the remaining ones. The final term is the number of bits required to identify the next patch I' to be recursively encoded. The recursion terminates in base case (2) where I must be encoded directly. In this work we use 24-bit color per pixel and thus $E_{\text{direct}}(I) = 24|\Omega(I)|$.

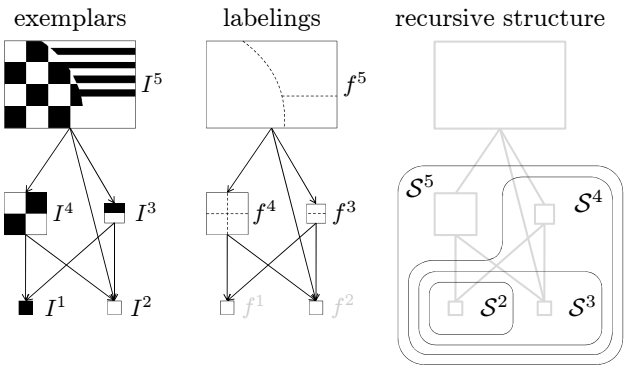

Figure 3. Illustration of how (1) represents image I leading to an RDAG over patches. Each recursive step k encodes patch I^k by finding a labeling f^k that optimally partitions I^k using patches from $\mathcal{S}^k = \{I^1, ..., I^{k-1}\}$. Here, f^5 partitions $I = I^5$ into three unique appearance models based on patches $\{I^2, I^3, I^4\} \subset \mathcal{S}^5$ and I^1 is only used indirectly through the encodings of I^3 and I^4.

The choice of E_{enc} is application-specific, and in this work we focus on encodings that find good image segmentations. Figure 3 depicts the recursive process of energy (1) for an optimal set $\mathcal{S}$ and how it yields a hierarchical representation for image $I = I^5$. Section 3.2 describes our segmentation-based representation in more detail.

A minimal description length for I can be found, in principle, by minimizing $\mathring{E}(\mathcal{S}; I)$ over all possible sets of patches $\mathcal{S}$. The problem we ultimately want to solve is

$$E(I) = \min_{\mathcal{S}} \mathring{E}(\mathcal{S}; I) + 2\lg|\mathcal{S}| + 1 \quad (3)$$

where $2\lg|\mathcal{S}| + 1$ is enough bits to identify $|\mathcal{S}|$. Our recursive energy is designed to favor sets that have no redundant patches (*i.e.* not used by any encoding). However, searching over all possible $\mathcal{S}$ is intractable. In practice we select from patches that were sampled from the image at various positions and scales.

3.2. Encoding via Segmentation & Exemplars

Our choice of $E_{\text{enc}}(f; \mathcal{S}, I)$ defines description length of image I of size $n \times n$ w.r.t. a set of patches $\mathcal{S}$ as

$$E_{\text{enc}}(f; \mathcal{S}, I) = 2 + 2\lg n + \min\Big\{ E_{\text{direct}}(I), \quad (4)$$

$$E_{\text{data}}(f; \mathcal{S}, I) + E_{\text{partition}}(f; \mathcal{S}) \Big\} \quad (5)$$

where labeling $f : \Omega(I) \rightarrow \mathcal{L}$ maps each image pixel to a discrete label. Each label $(e, t) \in \mathcal{L}$ identifies a patch-based *exemplar* I^e under some transformation rule indexed by $t \in T(e)$. Index t identifies a mapping $\Omega(I) \rightarrow \Omega(I^e)$ from the image coordinate space to the patch coordinate space. Each set of transformation rules $T(e)$ is specifically associated with patch I^e, and therefore $|\mathcal{L}| = \sum_e |T(e)|$. E_{enc} can choose either to encode I directly (4) using 24-bits per pixel, or to partition I into segments (5) defined by optimal labeling f where each segment is described by a patch from S. (One bit is paid to distinguish between these two options, and $1 + 2\lg n$ bits to encode the size of image I)

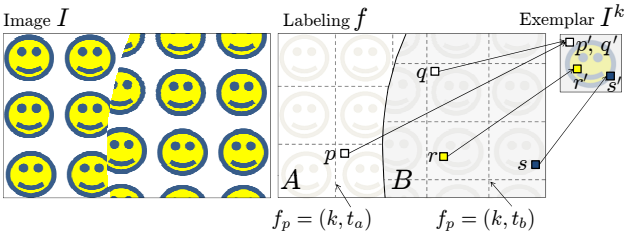

Figure 4. An example of mapping image pixels p to pixels p' in the coordinate space of patch I^k (size $n \times n$). Regions A and B are labeled by transformations t_a:$(x, y) \mapsto (x \bmod n, y \bmod n)$ and t_b:$(x, y) \mapsto (x + \frac{n}{2} \bmod n, y + \frac{n}{2} \bmod n)$ respectively. We refer to this class of transformations as *tilings*. Note that one patch and two tiling rules are enough to represent the image (left).

In the case where I is partitioned, E_{enc} chooses a labeling f that balances the complexity of the partition against the quality of the match between patches and the image. (Recall that the complexity of the set $\mathcal{S}$ itself is inherently minimized by (1).)

Data Cost Energy Given labeling f that maps each image pixel to a known patch pixel, the data cost energy E_{data} evaluates the number of additional bits required to reconstruct the original image w.r.t. the patch color values.

For each pixel $p \in \Omega(I)$ let its label be $f_p = (e_p, t_p) \in \mathcal{L}$ where e_p indexes patches from $\mathcal{S}$ and $t_p \in T(e_p)$ indexes one of transformation rules available for patch I^{e_p}. Figure 4 illustrates one possible class of such transformations we call 'tilings'; these are the only transformation rules we use, but others are possible (rotation, perspective). Let $I(p)$ denote the color at image pixel p and $I^{e_p}(t_p(p))$ be a corresponding color from the patch as determined by f_p. The total number of bits to encode I in terms of $\mathcal{S}$ given f is then

$$E_{\text{data}}(f; \mathcal{S}, I) = \sum_{p \in \Omega(I)} D(I(p) - I^{e_p}(t_p(p))) \qquad (6)$$

where $D(\cdot)$ is the number of bits to encode each difference.

The above definition of E_{data} is the simplest way to encode the color differences but, for natural images, incorporating non-rigid local deformations is necessary. Our experiments use a definition for E_{data} that allows mapped pixels to locally deviate from t_p and encode that deviation if it benefits the overall description length.

Partition Energy The partition energy evaluates the number of bits required to encode the labeling f itself, *i.e.* the partitioning of the image. We aim for labelings that are simple in the sense of being spatially coherent. This means f should be highly compressible w.r.t. some scanning order or neighborhood $\mathcal{N}(I)$ over the image pixels. For simplicity, we first define our partition energy w.r.t. to a simple 1D scanning pattern (e.g., zig-zag scan). However, for perceptual reasons suggested by Leclerc [12], in practice, we use multiple scanning patterns (resulting in 8-neighborhood) and average their expected description length.

The complexity of a partition depends directly on both the number of unique labels and the number of discontinuities $f_p \neq f_q$ for any $pq \in \mathcal{N}(I)$. Because each f_p has two components and $e_p \neq e_q \Rightarrow t_p \neq t_q$, there are two possible types of discontinuities: either the transformation rule changes, or both the patch *and* the transformation rule change (see faded and black lines in Figures 1, 2). We express the overall number of bits to encode partition f as

$$E_{\text{partition}}(f; \mathcal{S}) = E_{\text{lookup}}(f; \mathcal{S}) + \sum_{pq \in \mathcal{N}(I)} V(f_p, f_q, f) \qquad (7)$$

where V is the number of bits to encode a transition from label f_p to label f_q, and E_{lookup} represents the number of extra bits needed to know which patches and transformations are used in f. The effect of E_{lookup} is to make the description length V for transitions shorter and to encourage labelings that use fewer unique labels and are thus simpler.

Given the set of patches S and the labeling f, define $\hat{\mathcal{S}} = \{i \,|\, \exists e_p = i\}$ to be the set of patch indices actually used in the labeling f and similarly let $\hat{T}_i = \{j \in T(i) \,|\, \exists f_p = (i, j)\}$ be the indices of transformation rules actually used for each patch I^i. The smoothness cost V is then defined as the number of bits needed to describe a label transition

$$V(f_p, f_q, f) = \begin{cases} 2 + \lg|\hat{T}_{e_q}| + \lg|\hat{\mathcal{S}}| & e_p \neq e_q, t_p \neq t_q, \\ 2 + \lg|\hat{T}_{e_q}| & e_p = e_q, t_p \neq t_q, \\ 1 & \text{otherwise.} \end{cases} \qquad (8)$$

Since V depends only on the number of unique labels actually used in f, a lookup table is required to index the patches and transformations used. E_{lookup} represents the number of bits required for such a table and is defined as

$$E_{lookup}(f; \mathcal{S}) = |\hat{\mathcal{S}}| \lg|\mathcal{S}| + \sum_i |\hat{T}_i| \lg|T_i|. \qquad (9)$$

Note that our pairwise costs V depend on the orientation of a scanning path, *i.e.* $V(f_p, f_q, f) \neq V(f_q, f_p, f)$. In this case V is not submodular w.r.t α-expansion moves [3] and cannot be directly optimized with α-expansion. To address this we average pairwise costs over opposite orientations. It can be shown that for 1D scanning paths the minima of this modified energy correspond to minima of the original.

4. Optimizing Our Recursive Energy

In Sec. 4.1 we describe our greedy approach to optimize (3) . The problem is difficult, first because we cannot search over all possible $\mathcal{S}$ in practice, and second because even the key subproblem of minimizing E_{enc} inside each recursive step is $\mathcal{NP}$-hard. This subproblem requires minimizing a multi-label energy with label costs defined over very large subsets of labels. Unfortunately, the algorithm in [6] can easily get stuck in poor local minima for such energies. In Sec. 4.2 we introduce our hierarchical fusion algorithm to better optimize such energies in practice.

4.1. Greedy RDAG Construction Over Patches

Since we cannot search over all possible subsets of patches in (3) we restrict our search space to a set $\bar{\mathcal{S}}$ of patches sampled from the image, including the image itself. We adopt a greedy approach and sort all the patches in $\bar{\mathcal{S}}$ by ascending order of size. This greedy order determines the elimination order in (1), as illustrated in Fig. 3. To build the RDAG we add one patch at a time, and at step k of our greedy algorithm we compute the minimal description length of encoding I^k using patches from the set $\bar{\mathcal{S}}^k = \{I^1, \ldots, I^{k-1}\}, \bar{\mathcal{S}}^k \subseteq \bar{\mathcal{S}}$. We assume that patches $\{I^1, \ldots, I^{k-1}\}$ have already been encoded, their respective optimal partitions $\{f^1, \ldots, f^{k-1}\}$ computed, and therefore the dependencies between current patches in $\bar{\mathcal{S}}^k$ are fixed. Figure 5 illustrates a few steps of building the RDAG for the example and elimination order given in Fig. 3.

Finding an MDL representation for I^k entails choosing a good subset of patches from $\bar{\mathcal{S}}^k$. The main difficulty is that a patch $I^i \in \bar{\mathcal{S}}^k$ can be used both directly *and* indirectly (through its parent) to encode I^k, yet the cost of encoding I^i itself *must be paid at most once*. For example, consider step 5 in Fig. 5. The optimal labeling f^5 uses I^2 directly in one region, and indirectly (e.g. through I^3) in another. However, the cost of encoding I^2 must be paid even if only I^3 is used. This dependency cannot be represented by independent per-label costs for using I^2 and I^3 when computing f^5.

We use energies with label *subset* costs [6] to correctly account for the inter-patch dependencies at step k. These costs represent the description length in (1) computed by recursion on any possible subset of $\bar{\mathcal{S}}^k$. In order to define the necessary label subsets we begin with a formal description of the dependencies in the hierarchy of patches. We denote by $\mathcal{G}^k = \{\mathcal{V}^k, \mathcal{A}^k\}$ the dependecies in the RDAG at step k, with vertices $\mathcal{V}^k = \{v_1, \ldots, v_{k-1}\}$ corresponding to the patches in $\bar{\mathcal{S}}^k$. Whenever patch I^j depends directly on patch I^i for its encoding ($i < j < k$), we use an arc $(v_j, v_i) \in \mathcal{A}^k$ to denote this dependency. For each vertex $v_i \in \mathcal{V}^k$ we denote its graph predecessors and successors by sets $\mathcal{X}_i$ and $\mathcal{Y}_i$ respectively.

$$\mathcal{X}_i = \{\, j \mid v_i \text{ is reachable from } v_j\} \cup \{i\}$$
$$\mathcal{Y}_i = \{\, j \mid v_j \text{ is reachable from } v_i\}$$

$\mathcal{X}_i$ indexes all patches that (directly or indirectly) use the patch I^i for encoding. $\mathcal{Y}^i$ indexes all patches that are directly or indirectly used to encode patch I^i (Fig. 5, right).

Let $L_i = \{(e,t) \in \mathcal{L} \mid e \in \mathcal{X}_i\}$ be the set of labels in $\mathcal{L}$ corresponding to patches that depend on I^i for encoding. Our energy must be defined such that if any $f_p \in L_i$ then the cost of encoding patch I^i is paid. Denote by h_{L_i} the cost assigned to label subset L_i. At each step k of our greedy RDAG construction we optimize

$$\tilde{E}^k = \min_f E_{\text{enc}}(f; \bar{\mathcal{S}}^k, I^k) + \sum_{i=1}^{k-1} h_{L_i} \delta_{L_i}(f) \qquad (10)$$

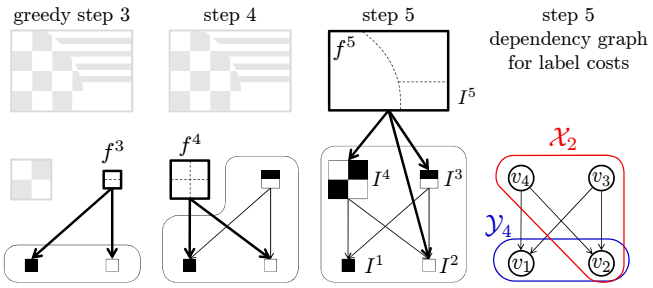

Figure 5. Steps k=3,4,5 of our greedy approach. At step 3, the patch I^3 is encoded using the set $\bar{\mathcal{S}}^3 = \{I^1, I^2\}$. For this trivial case, finding optimal f^3 in (10) is a two-label energy where the "label cost" of using each label is E_{direct}. After step 5, the final representation for patch I^5 depends directly on I^2, I^3, I^4 and indirectly on I^1. Note that I^2 is also used indirectly through the encoding of I^3. The rightmost column is explained in the text.

where the indicator function δ_{L_i} is defined on a label subset L_i as

$$\delta_{L_i}(f) = \begin{cases} 1 & \exists p : f_p \in L_i \\ 0 & \text{otherwise.} \end{cases}$$

The label costs in $\tilde{E}^k$ must be carefully designed to account for the specific dependency structure already present in $\mathcal{G}^k$. We define label costs as $h_{L_i} = \tilde{E}^i - \sum_{j \in \mathcal{Y}_i} h_{L_j}$. To understand why h_{L_j} are subtracted consider the following. First, when $\tilde{E}^i$ was computed it included (by definition) the cost of encoding all patches $I^j, j \in \mathcal{Y}_i$ on which it depends. This also means that $i \in \mathcal{X}_j \; \forall j \in \mathcal{Y}_i$ and the costs h_{L_j} will already be activated whenever the patch I^i is used. Hence, in order not to double count the costs $h_{L_j}, j \in \mathcal{Y}_i$ they should be subtracted from h_{L_i}.

Consider for example step 5 in Fig. 5. The label subsets that are used at this step are as follows:

$$\begin{array}{ll} \mathcal{X}_1 = \{1,3,4\} & \text{with } h_{L_1} = \tilde{E}^1 \\ \mathcal{X}_2 = \{2,3,4\} & \text{with } h_{L_2} = \tilde{E}^2 \\ \mathcal{X}_3 = \{3\} & \text{with } h_{L_3} = \tilde{E}^3 - h_{L_1} - h_{L_2} \\ \mathcal{X}_4 = \{4\} & \text{with } h_{L_4} = \tilde{E}^4 - h_{L_1} - h_{L_2} \end{array}$$

The costs h_{L_1} and h_{L_2} were subtracted from h_{L_3} and h_{L_4} because the sets L_1 and L_2 already include the patches I^3 and I^4. Without the subtraction, any labeling that uses both I^1 and I^3 together will pay h_{L_1} twice. Below we provide pseudo-code for greedy RDAG construction.

Greedy RDAG Construction

for $k = 1..|\bar{\mathcal{S}}|$
  $\bar{\mathcal{S}}^k = \bar{\mathcal{S}}^{k-1} \cup \{I^k\}, \quad \mathcal{V}^{k+1} = \mathcal{V}^k \cup \{v_k\}$
  set up label subsets L_i and label costs h_{L_i} using $\mathcal{G}^k$
  $f^k = \text{argmin}_f \tilde{E}^k$ // minimize with HF (Sec. 4.2)
  if $\tilde{E}^k(f^k)$ did not use $E_{\text{direct}}(I^k)$
    $\mathcal{A}^{k+1} = \mathcal{A}^k \cup \{(v_k, v_j) | \exists f_p^k = (j, \cdot)\}$
  create $\mathcal{X}_k, \mathcal{Y}_k$, update $\mathcal{X}_j \;\; \forall j < k$

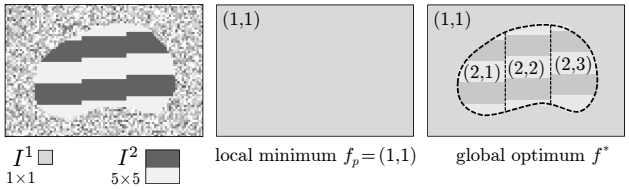

Figure 6. A local minimum for α-expansion with label subset costs [6]. Here the subset L_2 includes labels $(2,1),(2,2),(2,3)$. If the current labeling uses only patch I^1, the label cost h_{L_2} of introducing patch I^2 is too high for any one transformation $(2,t)$ to be worth expanding. Only by simultaneously expanding on three transformations of I^2 can we find an optimal labeling.

4.2. Hierarchical Fusion (HF) Algorithm

Since energy (10) has label costs, the natural approach to optimizing it would be the α-expansion-based method proposed in [6]. However, this method has poor optimality guarantees when label costs are defined over large label subsets and, in practice, gets stuck in local minima. Figure 6 illustrates an example of a problematic local minimum for our formulation using [6]. The fundamental problem faced by expansion is that no individual label (i,t) from a subset L_i is worth expanding due to high label cost h_{L_i}. Only by expanding on many labels from L_i *simultaneously* can the decrease in data costs compensate for paying high label cost h_{L_i}. As suggested by Fig. 6, this means that for patch i we need to effectively expand on all its labels $(i,t), t \in T(i)$ at once (*i.e.* all possible transformation rules).

We now describe our *hierarchical fusion* (HF) algorithm for optimizing the energy $\tilde{E}^k$ (10) at step k. Our algorithm begins by optimizing a set of independent sub-energies, one for each of $k-1$ patches (Fig. 7, bottom). Each sub-energy is defined by $\tilde{E}^k$ over a restricted set of labels and is optimized by running α-expansion [6] until convergence. For sub-energy j we restrict its set of labels to $\mathcal{L}^j=\{(j,\cdot)\}$, *i.e.* the transformation rules $T(j)$ available for patch I^j.

The result is a set $\mathcal{C}=\{c^1,\ldots,c^{k-1}\}$ of pixel labelings $c^j : \Omega(I^k) \rightarrow \mathcal{L}^j$. Each labeling c^j tries its best to 'reconstruct' the color data in I^k by transforming patch I^j via the rules in $T(j)$. The particular example in Fig. 7 shows how image I^5 is perfectly reconstructed by labeling c^4, but must pay heavy smooth costs to do so in regions that match I^4 poorly. In general, regions of I^k that locally match I^j will have low data costs (similar color) and low smooth costs (coherent regions with constant transformation). Regions that do not match will have either high data costs or high smooth costs. This fact is essential for the subsequent fusion step of our algorithm (Fig. 7).

Once labeling c^j is computed, each c^j_p designates a fixed transformation rule from $T(j)$ for pixel $p \in \Omega(I^k)$. The labelings in $\mathcal{C}$ are then stitched in a multi-label fusion energy still corresponding to (10). For each pixel, we define its label set for fusion as $\mathcal{L}^{\text{fuse}}_p=\{(j,c^j_p) \mid j \in 1...k-1\}$. In other words, by assigning label (j,c^j_p) to pixel p, the fusion

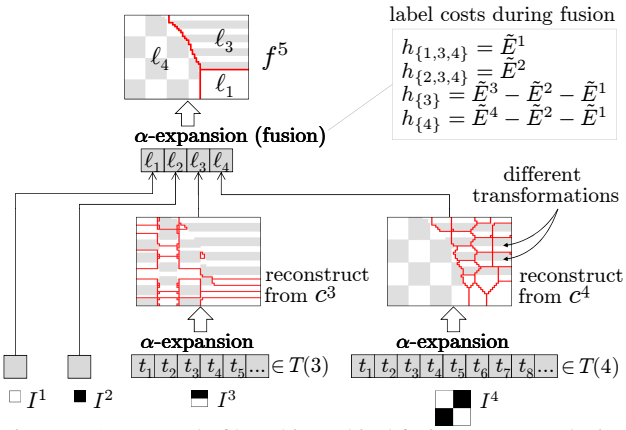

Figure 7. A portrayal of how hierarchical fusion computes the image representation in step 5 of Fig. 5. The upper-right region in labeling c^3 uses a constant transformation rule (a 'tiling') to synthesize a striped pattern. The fusion step therefore chooses ℓ_3 to represent this region in the encoding of I^5, and similarly chooses ℓ_4 to represent the checkerboard region.

step has chosen to represent p by patch j under transformation c^j_p. Optimizing over $\mathcal{L}^{\text{fuse}}$ drastically reduces the effective size of each label subset L_i in (10) from $\sum_{j\in\mathcal{X}_i}|T(j)|$ to $|\mathcal{X}_i|$. We thereby avoid local minima caused by subset costs h_{L_i} (Fig. 6). In summary, the fusion at step k chooses an MDL representation for image I^k by partitioning it and applying transformation rules to a good subset of patches from $\bar{\mathcal{S}}^k=\{I^1,\ldots,I^{k-1}\}$. This final partition determines the RDAG structure at step k (line **6** in Greedy RDAG).

HF Algorithm at step k († = α-exp. w/ label costs[6])

1 **for** $j = 1..k-1$
2 $\quad\mathcal{L}^j = \{(j,\cdot) \in \mathcal{L}\}$
3 $\quad c^j = \text{argmin}_c \tilde{E}^k,\ \ c : \Omega(I^k) \rightarrow \mathcal{L}^j$ †
4 $\mathcal{L}^{\text{fuse}}_p = \{(j,c^j_p) \mid j \in 1...k-1\}\ \ \forall p$
5 $f^k = \text{argmin}_f \tilde{E}^k,\ \ f : \Omega(I^k) \rightarrow \mathcal{L}^{\text{fuse}}$ †

One important detail in our MDL formulation is the dependence of V on the number of unique labels in f during each α-expansion. This is a question of V being as large as necessary according to (8), but no larger. Each invocation of α-expansion (lines **3**,**5**) begins with an underestimate of the number of unique labels. If the solution contains more unique labels than could be encoded by current V, we restart α-expansion with a higher estimate until we find a feasible solution. This process is guaranteed to terminate because increasing the estimate results in larger smooth costs and encourages fewer labels in the output.

5. Experiments

We collected a number of natural images and applied our method. As described in Sec. 4.1, we sample candidate patches from the image at various locations and scales. We eliminate redundant candidates by clustering the patches based on SSD of their Lab color values; this is done for

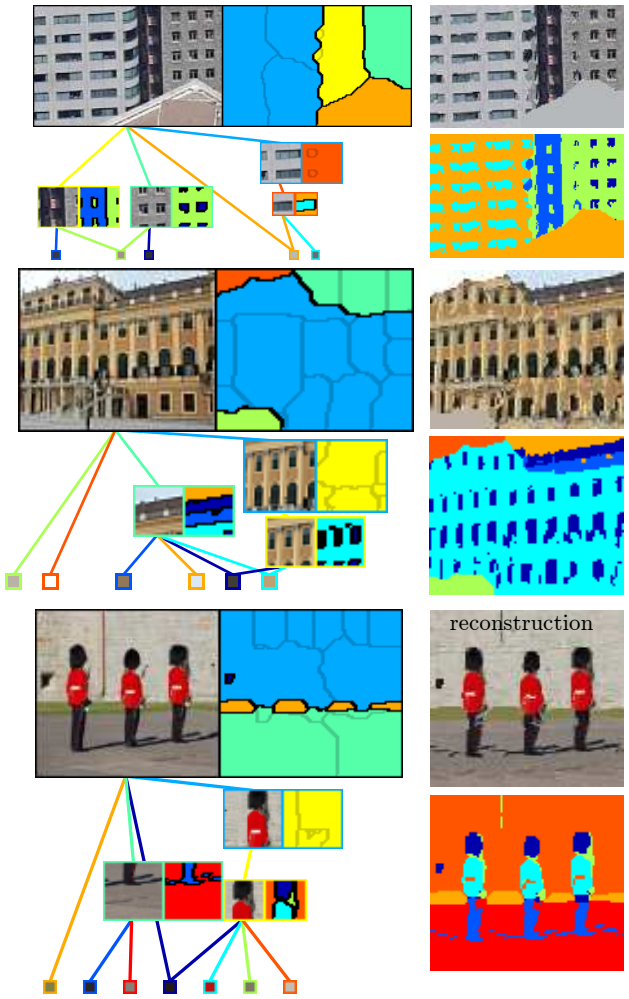

Figure 8. MDL image representations computed by our approach.

performance reasons only, as our energy prefers representations with no redundant patches and would not have selected them anyway. Figures 1,2,11 and 8 show results that are typical for images with repetitive structures. For each input image we show the dependency graph of patches along with their partitions and the top- and bottom-level segmentations. Notice in the top example of Fig. 8 that the left building is encoded by a patch of windows, which is in turn encoded by a single window, which is in turn encoded by solid colors (1-pixel patches). Also shown at top-right is a reconstruction of the image using only patch data, *i.e.* we drop the top-level information about color difference between patches and matching regions.

The top-level segmentation does not always follow object boundaries because our method has no semantic information about objects and therefore happily, and correctly, finds repetitive patterns along inter-object boundaries (*e.g.* roof/sky, soldier/wall). However, the low-level segmentations tend not to mix statistics between objects since they must be described by simpler patches. Fig. 9 shows how our bottom-level segmentations qualitatively compare

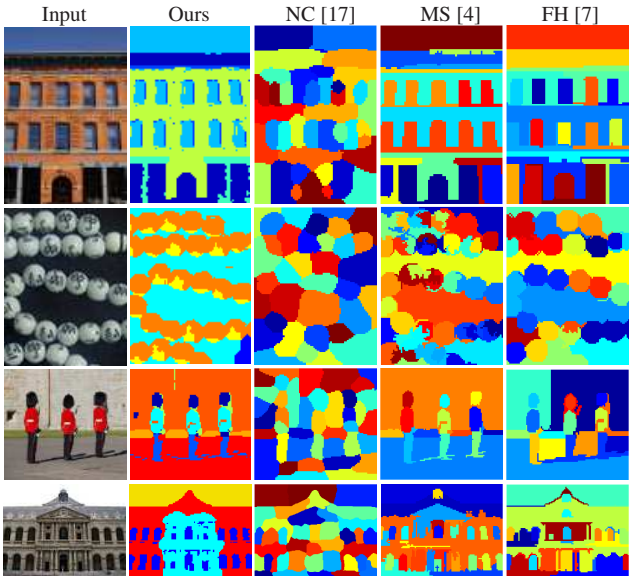

Figure 9. Our representation gives bottom-level segmentation where repeating elements tend to have the same appearance label, unlike methods not designed to detect repetition. For [17, 4, 7] we tuned parameters to give good segment sizes.

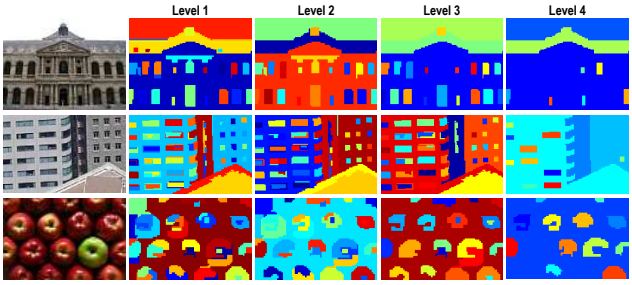

Figure 10. Four levels of a typical hierarchical region-merging approach [14]. Such methods are not designed to capture repetitive content and most of the merging happens in arbitrary order; this makes it difficult to know at which level (if any) the segmentation can be meaningfully analyzed. (Compare to ours in Figs.1,2,8)

to well-known superpixel methods. Fig. 10 shows how a standard region-merging method behaves on images with repetitive content.

6. Discussion

Our class of representations is not limited to patch-based appearance models. Hierarchies are a natural consequence of MDL when representing data by highly-complex models, and can be applied to other classes of appearance models (histograms, GMM, bag-of-features). Furthermore, more sophisticated transformations can easily improve results (reflections, rotations, or perspective transformations).

Though we introduced our HF algorithm in the context of automatic segmentation, we expect the scope of its applications to be much larger. The algorithm is designed to optimize energies that contain a hierarchy of label costs, and we foresee many applications for such energies.

Lastly, though this paper is not about compression, our

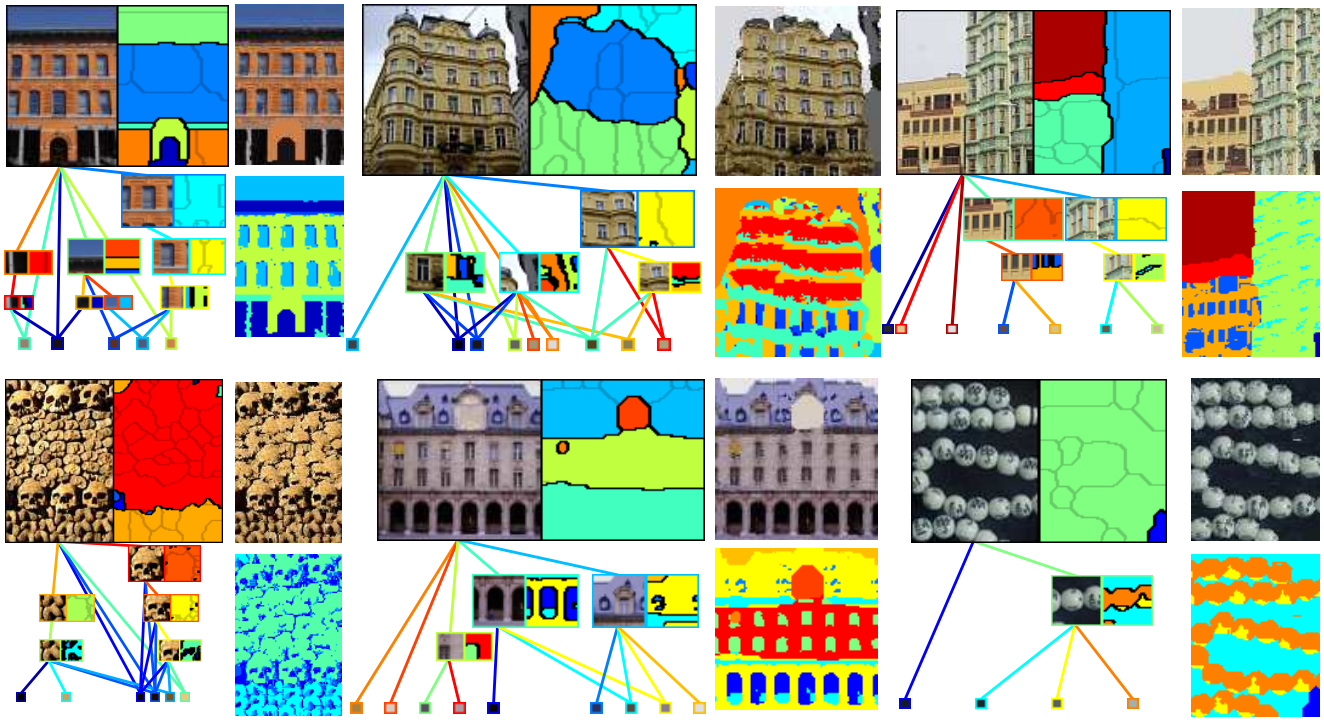

Figure 11. More results computed by our approach. Notice the dependencies at top-left: the largest patch (3 windows) is recursively encoded by repeating a smaller patch (1 window); this is shorter than encoding directly, and likewise for repetitive regions in the image.

representations can potentially be used to compress data with repetitive structures. For example, it is interesting to note that for the image shown in Fig. 8 top (taken from [19]), we obtain a compression rate of 8:1 for the lossy reconstruction shown and 3:1 for lossless representation, which is comparable to the rate reported in [19].

References

[1] S. Bagon, O. Boiman, and M. Irani. What is a Good Image Segment? A Unified Approach to Segment Extraction. In *ECCV*, October 2008.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *SIGGRAPH*, August 2009.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *TPAMI*, 2001.

[4] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *PAMI*, 24:603–619, 2002.

[5] A. Delong. *Advances in Graph Cut Optimization*. PhD thesis, University of Western Ontario, September 2011.

[6] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast Approximate Energy Minimization with Label Costs . In *CVPR*, June 2010.

[7] P. Felzenszwalb and D. Huttenlocher. Efficient Graph-Based Image Segmentation. *IJCV*, 59(2), 2004.

[8] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *ICCV*, 2009.

[9] J. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. In *ECCV*, May 2006.

[10] N. Jojic, B. J. Frey, and A. Kannan. Epitomic analysis of appearance and shape. In *CVPR*, 2003.

[11] M. P. Kumar and D. Koller. MAP estimation of semi-metric MRFs via hierarchical graph cuts. In *UAI*, June 2009.

[12] Y. G. Leclerc. Constructing simple stable descriptions for image partitioning. *IJCV*, 3(1):73–102, May 1989.

[13] V. Lempitsky, C. Rother, S. Roth, and A. Blake. Fusion moves for markov random field optimization. *TPAMI*, 32:1392–1405, August 2010.

[14] R. Nock and F. Nielsen. Statistical Region Merging. *TPAMI*, 26(11), November 2004.

[15] C. Rother, V. Kolmogorov, and A. Blake. GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. In *SIGGRAPH*, 2004.

[16] E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *CVPR*, 2001.

[17] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, Aug. 2000.

[18] D. Simakov, Y. Caspi, E. Shechtman, and M. Irani. Summarizing visual data using bidirectional similarity. In *CVPR*, June 2008.

[19] H. Wang, Y. Wexler, E. Ofek, and H. Hoppe. Factoring repeated content within and among images. *SIGGRAPH*, 2008.

[20] C. Wu, J.-M. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. In *ECCV*, 2010.

[21] R. Zabih and V. Kolmogorov. Spatially Coherent Clustering with Graph Cuts. In *CVPR*, June 2004.

[22] G. Zeng and L. V. Gool. Multi-label image segmentation via point-wise repetition. In *CVPR*, June 2008.

[23] S. C. Zhu and A. L. Yuille. Region competition: unifying snakes, region growing, and Bayes/MDL for multiband image segmentation. *TPAMI*, 18(9):884–900, 1996.