Fast Multiscale Image Segmentation

Eitan Sharon, Achi Brandt*, Ronen Basri†
Dept. of Applied Math
The Weizmann Inst. of Science
Rehovot, 76100, Israel

Abstract

We introduce a fast, multiscale algorithm for image segmentation. Our algorithm uses modern numeric techniques to find an approximate solution to normalized cut measures in time that is linear in the size of the image with only a few dozen operations per pixel. In just one pass the algorithm provides a complete hierarchical decomposition of the image into segments. The algorithm detects the segments by applying a process of recursive coarsening in which the same minimization problem is represented with fewer and fewer variables producing an irregular pyramid. During this coarsening process we may compute additional internal statistics of the emerging segments and use these statistics to facilitate the segmentation process. Once the pyramid is completed it is scanned from the top down to associate pixels close to the boundaries of segments with the appropriate segment. The algorithm is inspired by algebraic multigrid (AMG) solvers of minimization problems of heat or electric networks. We demonstrate the algorithm by applying it to real images.

1 Introduction

Image segmentation is a process of grouping together neighboring pixels whose properties (e.g., intensity values) are coherent. The resulting regions may indicate the presence of objects or parts of objects, and may be verified (or modified) later following a top-down analysis of the image and recognition. It is important to construct algorithms for segmentation that are efficient and that can faithfully extract regions of different sizes from an image. In this paper we introduce a fast graph algorithm for segmentation that finds an approximate solution to normalized cut measures and whose runtime is linear in the size of the image. In just one pass the algorithm provides a complete hierarchical decomposition of the image into segments.

A large class of graph algorithms have been adapted to deal with the segmentation problem. These algorithms typically construct a graph in which the nodes represent the pixels in the image and arcs represent affinities ("couplings") between nearby pixels. The image is segmented by minimizing a cost associated with cutting the graph into subgraphs. In the simpler version, the cost is the sum of the affinities across the cut [20]. Other versions normalize this cost by dividing it by the overall area of the segments [6] or by a measure derived from the affinities between nodes within the segments [17, 13, 19]. Normalizing the cost of a cut prevents over-segmentation of the image. Polynomial methods for finding a globally optimal solution when the cost is normalized exist when the graph is planar, but the runtime complexity of these methods is $O(N^2 \log N)$, where N denotes the number of pixels in the image (see [14, 6]). When the graph is non-planar the problem of finding a globally optimal solution is NP-hard. Therefore, approximation methods are employed. The most common of these uses spectral techniques to find an approximate solution. These spectral methods are analogous to finding the principal modes of certain physical systems. With these methods, and exploiting the sparseness of the graph, a cut can be found in $O(N^{3/2})$ [17].

Below we introduce a fast algorithm for segmentation. Our algorithm too finds an approximate solution to a normalized cut problem, but it does so in time that is linear in the number of pixels in the image with only a few dozen operations per pixel. Since a typical image may contain several hundreds of thousands of pixels, the factor $\sqrt{N}$ gained may be quite significant. The algorithm is based on representing the same minimization problem at different scales, enabling fast extraction of the segments that minimize the optimization criterion. Because of its multiscale nature, the algorithm provides a full hierarchical decomposition of the image into segments in just one pass. In addition, it allows us to modify the optimization criterion with scale so that we can incorporate higher order statistics of the segments when their size is sufficiently large to allow reliable extraction of such statistics. Our algorithm relates to the same physical systems whose modes are found by the spectral methods, but uses modern numeric techniques that provide a fast and accurate solution to these problems. The results of running our algorithm on a variety of images are at least comparable to the results obtained by the spectral methods.

Our algorithm proceeds as follows. Given an image, we first construct a graph so that every pixel is

*Research supported in part by Israel Ministry of Science Grant 4135-1-93 and by the Gauss Minerva Center for Scientific Computation.

†Research supported in part by the Unites States-Israel Binational Science Foundation, Grant No. 94-00100.

a node in the graph and neighboring pixels are connected by an arc. A weight is associated with the arc reflecting the likelihood that the corresponding pixels are separated by an edge. To find the minimal cuts in the graph we recursively coarsen the graph using a *weighted aggregation* procedure in which we repeatedly select smaller sets of representative pixels (*blocks*). These representative pixels do not have to lie on a regular grid, giving rise to an irregular pyramid. The purpose of these coarsening steps is to produce smaller and smaller graphs that faithfully represent the same minimization problem. In the course of this process segments that are distinct from their environment emerge and they are detected at their appropriate size scale. After constructing the entire pyramid we scan the pyramid from the top down performing relaxation sweeps to associate each pixel with the appropriate segment.

In the simple version of our algorithm the couplings between block pixels at a coarse level are computed directly from the couplings between finer level pixels. In a variation of this algorithm we modify the couplings between block pixels to reflect certain global statistics of each block. These statistics can be computed recursively throughout the coarsening process and may include the average intensity level of the blocks, the position of their center, their principal orientation, their area, texture measurements, etc. This enables us, for example, to identify large segments even if the intensity levels separating them vary gradually.

Our algorithm is inspired by *Algebraic Multigrid* (AMG) solvers applied to physical systems of heat or electric networks. By analogy the graph produced from the image to be segmented can be thought of as such a network, where the couplings between the intensity levels of neighboring pixels are treated as conductivity measures. A common problem is to find the optimal state of such a network given a set of constraints, which physically represent a set of heat sources or input currents. AMG solvers provide a fast, multiscale way to solve such optimization problems. For our purposes we do not need to minimize the energy for any particular set of constraints. Instead, it will be sufficient to borrow from the AMG solver its weighted aggregation (or interpolation) rules, which are independent of the particular constraints. While doing so we solve a problem of a different nature than that traditionally solved by AMG.

Pyramidal structures have been used in many algorithms for segmentation (see reviews in [7, 9, 11]). However, methods that use regular pyramids (e.g., [10]) have difficulties in extracting regions of irregular structures. Methods that construct irregular pyramids (e.g., [1, 5, 12, 18]) are strongly affected by local decisions. Fuzzy C-means clustering algorithms (e.g., [4]) avoid such premature decisions, but they involve a slow iterative process. Also related are algorithms motivated by physical processes (e.g., [8, 15]).

The paper is divided as follows. Section 2 formulates the segmentation problem and describes the principles of our method. Section 3 describes the algorithm. Section 4 discusses how more global properties of segments can be incorporated in the algorithm. Section 5 discusses the computational complexity of the algorithm. Finally, Section 6 provides experimental results.

2 Motivation and Formulation

In this section we cast the segmentation problem as a graph clustering problem (Sec. 2.1). Then, we describe the coarsening process (Sec. 2.2). Finally, we interpret this as an aggregation process (Sec. 2.3).

2.1 Problem Definition

Given an image Ω that contains $N = n \times n$ pixels we construct a graph in which each node represents a pixel and every two nodes representing neighboring pixels are connected by an arc. In our implementation we connected each node to the four neighbors of the respective pixel, producing a planar graph. (Note that the method we present can be applied also to non-planar graphs. In fact, the graphs obtained following the coarsening steps are non-planar.) Below we denote a pixel by an index $i \in \{1, 2, ..., N\}$ and its intensity by g_i. To every arc connecting two neighboring pixels i and j we assign a positive *"coupling"* value a_{ij}, reflecting the degree to which they tend to belong to the same segment. For example, a_{ij} could be a decreasing function of $|g_i - g_j|$. In our implementation we used local responses to edge filters to determine the couplings between elements (see Section 3).

To detect the segments, we associate with the graph a *state vector* $u = (u_1, u_2, ..., u_N)$, where $u_i \in \mathbb{R}$ is a *state variable* associated with pixel i. We define a segment $S^{(m)}$ as a collection of pixels, $S^{(m)} = \{i_{m_1}, i_{m_2}, ..., i_{m_{n_m}}\}$ and associate with it a state vector $u^{(m)} = (u_1^{(m)}, u_2^{(m)}, ..., u_N^{(m)})$, in which

$$u_i^{(m)} = \begin{cases} 1 & \text{if} \quad i \in S_m \\ 0 & \text{if} \quad i \notin S_m \end{cases} . \tag{1}$$

In practice, we allow the state variables to take non binary values. In particular, we expect that pixels near fuzzy sections of the boundaries of a segment may have intermediate values $0 < u_i^{(m)} < 1$ reflecting their relative tendency to belong to either the segment or its complement.

Next, we define an energy functional to rank the segments. Consider first the functional

$$E(u) = \sum_{<i,j>} a_{ij}(u_i - u_j)^2 , \tag{2}$$

where the sum is over all pairs of adjacent pixels i and j. Clearly, for an ideal segment (with only binary state variables) $E(u^{(m)})$ sums the coupling values along the boundaries of $S^{(m)}$. With such a cost function small segments (and similarly very large ones) are often encouraged. To avoid such preference we can modify this energy as follows:

$$\Gamma(u) = E(u)/V^{\alpha}(u) , \tag{3}$$

where $V(u)$ denotes the "volume" of the respective segment, $V(u) = \sum_i u_i$, and α is some predetermined parameter. Thus, for example, $V(u^{(m)})$ will measure the area in pixels of $S^{(m)}$. To avoid selecting very large segments we consider only segments whose total volume is less than half of the entire image. This is equivalent to defining the volume as $min\{V(u), N-V(u)\}$. Alternatively, we can replace the volume by the product $V(u)(N-V(u))$. This and similar modifications of $\Gamma(u)$ can too be incorporated in our fast algorithm.

Note that setting $\alpha = 0.5$ will eliminate size preference since $\Gamma(u^{(m)})$ in this case is roughly the average of the couplings along the boundary of $S^{(m)}$. ($E(u^{(m)})$ is the sum of the couplings along the boundary of $S^{(m)}$, and $\sqrt{V(u^{(m)})}$ is roughly proportional to the perimeter of $S^{(m)}$.) In contrast, setting $\alpha > 0.5$ will create preference for large segments. In our implementation we used $\alpha = 1$, which is equivalent to the so called "average" or "normalized" cut measures (e.g., [6, 16, 17]).

Finally, the volume of u can be generalized by replacing $V(u)$ by

$$V_\phi(u) = \sum_{i=1}^{N} \phi_i u_i \ , \quad \sum_{i=1}^{N} \phi_i = N \ , \tag{4}$$

where ϕ_i is a "mass" assigned to the pixel i. This will become important in coarser steps when nodes may draw their mass from sets of pixels of different size. Also, in the finest scale we may assign lower volumes to pixels at "less interesting" or "less reliable" parts of the image, e.g., along its margins.

2.2 Problem Coarsening

We now present a method for the recursive step by step *coarsening* of the segmentation problem. In each coarsening step a new, approximately equivalent segmentation problem will be defined, reducing the number of state variables to a fraction (typically between 1/4 and 1/2) of the former number. We construct the coarser problems such that each of the coarse variable will represent several fine variables with different weights, and every fine variable will be represented by several coarse variables with different weights. The low-energy configurations of the coarse problem will reflect the low-energy configurations of the fine problem.

Below we describe the *first* coarsening step. The state variables in the coarser problem can be thought of as the values (ideally 0 or 1) of a *diluted* set of pixels, i.e., a subset C of the original set of pixels. The values u_i associated with the rest of the pixels ($i \notin C$) will be determined from the coarse state variables using pre-assigned dependence rules. These rules will define $\Gamma(u)$ as a functional of the smaller set of variables, i.e., $\Gamma^c(\{u_i\}_{i\in C})$. We shall select C and the dependence rules so that the detection of segments with small Γ^c (in the coarser problem) would lead to segments with small Γ (in the fine problem).

Generally, for *any* chosen subset of indices

$$C \stackrel{def}{=} \{c_k\}_{k=1}^{K} \subset \{1, 2, ..., N\} \ ,$$

denote u_{c_k} as U_k, we will choose dependence rules of the form of a weighted interpolation rule:

$$u_i = \sum_{k=1}^{K} w_{ik} U_k \ , \tag{5}$$

where $w_{ik} \geq 0$, $\sum_{k=1}^{K} w_{ik} = 1$, and for $i = c_k \in C$, $w_{ik} = 1$. We will consider only *local* interpolation rules, i.e., $w_{ik} = 0$ for all pixels c_k not in the neighborhood of pixel i. The values of w_{ik} will be determined by the coupling values only, and will not depend on the values of the state variables (see below).

Substituting (5) into (2) we get

$$E^c(U) \stackrel{def}{=} E(u) = \sum_{k,l} A_{kl}(U_k - U_l)^2 \ , \tag{6}$$

where the couplings A_{kl} between the coarse-level variables are given by

$$A_{kl} = \sum_{i \neq j} a_{ij}(w_{jl} - w_{il})(w_{ik} - w_{jk}) \ . \tag{7}$$

In addition, substituting (5) into (4) we get

$$V^c(U) \stackrel{def}{=} V_\phi(u) = V_\Phi(U) = \sum_{k=1}^{K} \Phi_k U_k \ , \tag{8}$$

where

$$\Phi_k = \sum_i \phi_i w_{ik} \ , \quad k = 1, ..., K \ . \tag{9}$$

Thus, the dependence rules (5) yields

$$\Gamma^c(U) \stackrel{def}{=} \Gamma(u) = E^c(U) / \left[V^c(U)\right]^\alpha \ . \tag{10}$$

The set C itself will be chosen in such a way that each pixel $i \notin C$ is *strongly coupled to pixels in* C. By this we mean roughly that

$$\sum_{c_k \in C} a_{ic_k} \geq \beta \sum_j a_{ij} \ , \tag{11}$$

where β is a control parameter. (A somewhat weaker type of requirement emerges in the Appendix.) This choice will ensure that for *any* low-energy configurations the values of u indeed depend, to a good approximation, on those of the subset U. This choice of C is common in applying fast, multiscale AMG solvers (e.g., [3]).

We now discuss the interpolation rule in Eq. (5). Given a segment S_m, we define $U^{(m)}$ as

$$U_k^{(m)} = \begin{cases} 1 & \text{if} \quad c_k \in S_m \\ 0 & \text{if} \quad c_k \notin S_m \ , \end{cases} \tag{12}$$

and define $\tilde{u}^{(m)}$ as the configuration interpolated from $U^{(m)}$ by using Eq. (5). That is,

$$\tilde{u}_i^{(m)} = \sum_{k=1}^{K} w_{ik} U_k^{(m)} \ . \tag{13}$$

Note that $E^c(U^{(m)}) = E(\tilde{u}^{(m)})$, $V^c(U^{(m)}) = V(\tilde{u}^{(m)})$, and hence $\Gamma^c(U^{(m)}) = \Gamma(\tilde{u}^{(m)})$. A *proper* interpolation rule should satisfy the condition that for every S_m, $\Gamma^c(U^{(m)}) = \Gamma(\tilde{u}^{(m)})$ is small if and only if $\Gamma(u^{(m)})$ is small.

One possible interpolation rule could be that a state variable $\tilde{u}_i$ for $i \notin C$ would inherit its state from the coarse state variable U_k to which it is most strongly attached (in other words, $\tilde{u}_i = U_k$ such that a_{ik} is maximal). This rule, however, may lead to mistakes in assigning the correct state to the interpolated variables due to nearby outliers, which in turn may result in a noticeable increase in the energy $E(\tilde{u}^{(m)})$ associated with the segment. Consequently, the minimization problem with the coarse variables will poorly approximate the minimization problem with the fine variables.

Instead, we will set the interpolation weights as follows:

$$w_{ik} = \frac{a_{ic_k}}{\sum_{l=1}^{K} a_{ic_l}} \ , \quad \forall i \notin C \ , c_k \in C \ . \tag{14}$$

These settings are commonly used by the AMG minimizer [3]. (For a definition of weights that leads to an even more precise interpolation - see the Appendix.) With this interpolation rule the state of a variable $\tilde{u}_i$, $i \notin C$, is determined by several nearby coarse pixels with pixels coupled more strongly affecting its value more.

It is straightforward to verify that boundary sections of a segment across which intensity variations are sharp contribute very little to the energy associated with the segment, whereas sections of the boundary across which intensity is varying gradually contribute most of the energy of the segment. It can be shown further that when the problem is coarsened the contribution of such sections in general decreases by about half. Since the volume of a segment is roughly preserved when the problem is coarsened, we obtain that for a segment S_m that is distinct from its surrounding $\Gamma^c(U^{(m)}) \approx \Gamma(u^{(m)}) \approx 0$, whereas for a segment S_m that is not strongly decoupled along its boundaries $\Gamma^c(U^{(m)}) \approx \frac{1}{2}\Gamma(u^{(m)})$. Thus, under the weighted interpolation (14), the problem of finding all segments S_m for which $\Gamma(u^{(m)})$ is below a certain threshold is equivalent approximately to the smaller, *coarse* problem of finding all S_m for which $\Gamma^c(U^{(m)})$ is below *half* the same threshold.

Note that the resulting coarse problem is exactly of the same form as the original problem, and hence it can in turn be reduced using the same procedure to an equivalent, yet coarser problem of the same form. This recursive coarsening process is terminated when the number of variables is sufficiently small so that the problem can be solved directly for the coarsest grid.

There is one case in which a state variable cannot be approximated accurately by the state variables of its neighbors. This happens when a salient segment S_m coincides at some scale with a single pixel i; i.e., $u_i^{(m)} = 1$ while $u_j^{(m)} = 0$ for $j \neq i$. (This, of course, would not happen usually at the original, finest level, but at coarser levels of the algorithm, where "pixels" are no longer original image pixels.) Consequently, if $i \notin C$ then the segment will no longer be represented at the coarser levels. But it is exactly at this point of the coarsening process that we can detect that $\Gamma(u^{(m)})$ is small, and hence identify the salient S_m in its natural size scale (see algorithm in Section 3).

2.3 Hierarchical Aggregation

A natural and useful way to interpret each coarsening step is as an *aggregation* step. In that view we are choosing small *aggregates* of pixels, in terms of which the minimization problem can be reformulated with a substantially smaller number of variables. That is, enumerating the aggregates 1,2,...,K, we associate with the k-th aggregate a *"block variable"* U_k, and we derive from the original minimization problem a minimization problem in terms of U_1,...,U_K[1].

The interpolation rule that relates the coarse to the fine pixels ((5) and (14)) leads to a process of *weighted* aggregation, in which a *fractions* w_{ik} of a pixel i can be sent into the aggregate k. This fraction may be interpreted as the *likelihood* of the pixel i to belong to the aggregate k. These likelihoods will then accumulate and reinforcing each other at each further coarsening step.

The choice of the coarser aggregates and the nature of this coarsening process is such that strongly coupled aggregates join together to form yet coarser aggregates. A set of pixels with strong internal couplings but with weak external couplings is bound to result at some level of coarsening in one aggregate which is weakly coupled to all other aggregates of that level. Such an aggregate will indicate the existence of an image segment (see Sec. 3).

The "coarse couplings" relations (Eq. (7)) can be somewhat simplified, yielding a similar coarsening process, named *Iterated Weighted Aggregation (IWA)*. IWA consists of exactly the same steps as the AMG coarsening, except that the coarse couplings $\{A_{kl}\}$ are calculated by the simpler formula

$$A_{kl} = \sum_{i \neq j} w_{ik} a_{ij} w_{jl} \ . \tag{15}$$

It can be shown that (15) in many situations provides a good approximation to (7). In certain cases the two processes are identical, e.g., in the case that each pixel is associated with only two blocks. Moreover, (15) can be motivated by itself: it states that the coupling between two blocks is the sum of the couplings between

[1] The coarse variables in fact do not have to be identified each with a particular pixel, as in Sec. 2.2. Instead, they can be identified with weighted averages of pixels. But this generality does not improve the performance of the algorithm and is certainly less convenient.

the pixels associated with these blocks weigthed appropriately.

3 The Algorithm

Based on these ideas we have developed a segmentation algorithm that is composed of two stages. In the first stage salient segments are detected and in the second stage the exact boundaries of the segments are determined. The rest of this section describes the two stages.

3.1 Detecting the Salient Segments

Given an image we consider each pixel to be a node connected to its four immediate neighbors. We then assign coupling values between each pair of neighbors. The coupling values a_{ij} are set to be $a_{ij} = \exp(-\mu r_{ij})$, where μ is a global parameter, and r_{ij} is an "edgeness" measure between i and j. Specifically, for horizontally spaced neighbors i and j we tested the presence of an edge in five orientations at the angular range $-45° \leq \theta \leq 45°$ about the vertical direction, each by differentiating two 3×1 masks whose centers are placed on i and j. We then took r_{ij} to be the maximal of the five responses.

Next, we coarsen this graph by performing iterated weighted aggregation. At each step of the coarsening we first select block pixels and then update the couplings between the blocks. Subsequently, we obtain a pyramidal structure that makes the optimal segments explicit.

Selecting the block pixels. We first order the nodes (pixels) by the volume they represent. (We sort the nodes by bucketing to maintain linear runtime complexity, see Sec. 5.) We select the first pixel to be a block. Then, we scan pixels according to this order and check their degree of attachment each to the previously selected blocks. Whenever we encounter a pixel that is weakly attached to the selected blocks we add that pixel to the list of blocks.

Specifically, let $C^{(i-1)}$ denote the set of blocks selected before a pixel i is tested, we check the inequality

$$\max_{j \in C^{(i-1)}} a_{ij} \geq \tilde{\alpha} \sum_{l} a_{il} \ , \tag{16}$$

where $\tilde{\alpha}$ is a parameter (typically $\tilde{\alpha} \sim .1$). Note that since generally a node is connected to a small number of neighbors it must be coupled strongly to at least one of its neighbors. In case the inequality is satisfied we set $C^{(i)} = C^{(i-1)}$, otherwise we set $C^{(i)} = C^{(i-1)} \bigcup \{i\}$. As a result of this process almost every pixel $i \notin C$ becomes strongly coupled to the pixels in C. The few remaining pixels are then added to C.

Segmentation. We update the couplings between the blocks using Eq. (15), where the weights w_{ik} are defined by (14) (or its generalization described in the Appendix). In addition, we compute the volume Φ_k of each block at this level using Eq. (9). Next, we want to determine if a block represents a salient segment. The saliency of a segment is given by the ratio between the sum of its external couplings and its volume. When we compute the saliency of a block, however, we need to take into account that every coarsening step diminishes the external couplings of the segment by about a half. We can compensate for this reduction by multiplying this ratio by 2 to the power of the level number. Thus, the saliency of a block k becomes

$$\Gamma(U_k) = \frac{\sum A_{kl}}{\Phi_k^{\alpha}} 2^{\sigma} \ ,$$

where σ denotes the scale. Alternatively, we can use the volume of the block as a measure of scale, in which case we obtain

$$\Gamma(U_k) = \frac{\sum A_{kl}}{\Phi_k^{\alpha-\gamma}} \ ,$$

where γ can be set between 0.5 to 1 according to the ratio of pixels that survive each coarsening step (0.25 to 0.5 respectively). In our implementation we simply compare the blocks of the same scale and detect the ones whose saliency values are very low. We then allow these blocks to participate in forming larger blocks to obtain a hierarchical decomposition of the image into segments.

3.2 Sharpening Segment Boundaries

During the first stage of our algorithm a salient segment is detected as a single element at some level of the pyramid. It remains then to determine exactly which pixels of the original image (at the finest level) in fact belong to that segment. One way to determine which pixels belong to a segment is to compute recursively the degree of attachment of every pixel to each of the blocks in the pyramid. Unfortunately, the degrees of attachment computed this way will often produce "fuzzy" values between 0 to 1 particularly near the boundaries of a segment, rendering the decision of the extent of a segment somewhat arbitrary. To avoid this fuzziness we scan the pyramid from coarse to fine starting at the level in which a segment is detected and apply relaxation sweeps whose intent is to sharpen the boundaries of a segment. Below we describe one step of the algorithm.

Suppose a segment S_m has been detected, and suppose that at a certain level (which we will call now the "coarse-level") we have already determined which pixels belong to S_m, we show how to determine at the next finer level (called now the "fine level") which pixels belong to S_m. Using the same notation as before, the coarse level variables, $\{U_k^{(m)}\}_{k=1}^{K}$, satisfy (12). Actually, along the boundaries of S_m some $U_k^{(m)}$'s may assume values between 0 and 1. Our task is to determine which pixels $\{u_j^{(m)}\}_{j=1}^{N}$ satisfy (1), but again allowing only pixels along the boundaries to obtain intermediate values between 0 and 1. Guided by the principle of minimizing $\Gamma(u^{(m)})$, a *sharpening cycle* consists of the following steps, iteratively changing $\tilde{u}^{(m)}$.

We fix two parameters $0 < \delta_1 < \delta_2 < 1$ and define $D_{x,y}$ to be the set of all pixels i such that $x < \tilde{u}_i^{(m)} < y$ at the beginning of the cycle. We then modify $\tilde{u}^{(m)}$

by setting $\tilde{u}_i^{(m)} = 0$ for $i \in D_{0,\delta_1}$, setting $\tilde{u}_i^{(m)} = 1$ for $i \in D_{\delta_2,1}$, and leaving $\tilde{u}_i^{(m)} \in D_{\delta_1,\delta_2}$ unchanged. This is followed by applying ν "Gauss-Seidel relaxation sweeps" over $D_{\delta_1\delta_2}$, where ν is another free parameter. Each such "relaxation sweep" is a sequence of steps aimed at lowering $E(\tilde{u}^{(m)})$. In each sweep we go over all the pixels in $D_{\delta_1\delta_2}$, in any order. For each pixel i we replace $\tilde{u}_i^{(m)}$ by the new value $\sum_j a_{ij}\tilde{u}_j^{(m)}/(\sum_j a_{ij})$, which is the value for which $E(\tilde{u}^{(m)})$ is lowered the most. Since the volume $V(\tilde{u}^{(m)})$ is only marginally affected also $\Gamma(\tilde{u}^{(m)})$ is lowered. Since in the beginning of this procedure already only pixels around the boundaries have fuzzy values (because this procedure has been applied to the coarser level) this relaxation procedure converges quickly. Hence, a small number of sweeps, ν, will generally suffice. In our experiments we applied two relaxation sweeps in every level with, e.g., $\delta_1 = 1 - \delta_2 = .15$ in the first cycle and $\delta_1 = 1 - \delta_2 = .3$ in the second cycle. The final $\tilde{u}^{(m)}$ is defined as the desired vector $u^{(m)}$.

4 Modified Coarse Couplings

In the algorithm described above the couplings at all levels are derived directly from the couplings between the pixels at the finest level. However, since each element at a coarse level represents an aggregate of pixels we may use information about the emerging segments that is not directly available at the finest level to facilitate the segmentation process. We can thus measure "*observables*" at the coarse levels, and use them to increase or decrease the couplings between blocks obtained with the original algorithm. An example for such an observable is the average intensity of a block, which can be used to separate segments even when the transition between their intensity values is gradual, and so they are difficult to separate at the finest levels. The average intensity G_k of a block k in the above coarsening step (Sec. 2.2) is defined as $G_k = \sum_i w_{ik} g_i / \sum_i w_{ik}$, where g_i denotes the intensity of pixel i; This observable can be calculated recursively at all coarser levels. Then, the couplings A_{kl} computed by (15) may be replaced, e.g., by $A_{kl} \exp(-\mu\, |G_k - G_l|)$, where μ is some predetermined constant.

The number of observables per aggregate can increase at coarser levels. Other possible observables include the center of mass of a block, its diameter, principal orientations, texture measures, etc. Using these observables it is possible to incorporate quite elaborate criteria into the segmentation process. For example, strong couplings can be assigned between two aggregates whose orientations align with the direction of the line connecting their centers of mass (or when their boundaries co-align), even when these aggregates are separated by a gap and thus do not inherit any mutual couplings from finer levels.

5 Computational Complexity

At every coarsening step we select a subset of the nodes such that the remaining nodes are coupled strongly to at least one of the nodes. Following this selection procedure almost no two neighboring nodes can survive to the next level. Thus, at every level of scale we obtain about half the nodes from the previous level. The total number of nodes in all levels, therefore, is about twice the number of pixels.

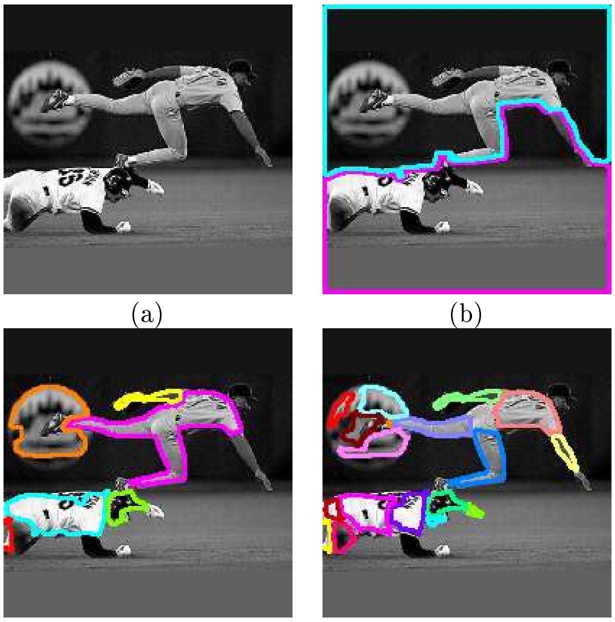
During the selection procedure there are two operations whose naive implementation may result in a nonlinear complexity. First, we need to order the nodes, say, according to their volumes. This can be done in linear time by dividing the range of possible volumes into a fixed number of buckets since it is unnecessary to sort nodes whose volumes are similar. Furthermore, in the first few levels the nodes usually have similar volumes, and so we do not apply this ordering. Instead, we merely scan the nodes in some arbitrary order. Secondly, for every node we need to find its maximal connection to the selected blocks (Eq. (16)). This operation can be implemented efficiently by noticing that every node need only to consider its neighboring nodes, typically up to 8 nodes. Finally, computing the degree of attachment of the pixels to all the block variables can be done in one pass once the pyramid is complete.

The number of operations per pixel can be reduced significantly by replacing the first 1-3 coarsening steps by equivalent geometric coarsening. In these coarsening steps the same operations are performed, but the pixels selected as blocks are determined in advance to lie along a regular grid of twice the meshsize. (This may require adding some of the fine pixels to the coarse set to avoid inaccurate interpolations.) With this modification it is possible to reduce the execution time of the algorithm to only *several dozen operations per pixel.*

In the following section we show examples of segmentation obtained with our implementation of the algorithm. The implementation is far from optimized, and, for example, we did not include geometric coarsening to reduce the number of operations per pixels. Due to wasteful space management, which lead to considerable page swapping, our implementation (written in C and run on an Intel 400MHz Pentium II processor) took 60 seconds to segment a 200×200 image. The pyramid produced in this run contained about 73000 nodes (less than twice the number of pixels.) Segmenting a 100×100 image took only 12 seconds.

6 Experiments

The following pictures demonstrate the application of our algorithm to several real images. Figure 1 shows an input image (adopted from [17]). At the top most scale the picture was divided into two segments. At scale 8 five segments stood out, two capturing most of the bodies of the two players, one captures the hand of one of the players, and one captures the head of the other. At scale 7 smaller segments are obtained, separating some of the body parts of the two players. Figure 2 was decomposed at level 10 into four segments, one of which captures the lioness. At level 8 the bottom segment was further decomposed into three segments, splitting the cub and the stone from the ground. Figure 3 shows the three segments obtained at scale 10, capturing the skies, the grass, and



(a) (b) (c) (d)

Figure 1: Segmentation results: (a) The input image. (b) Segments extracted at scale 11 (the boundaries of the segments are highlighted with color). (c) Scale 8. (d) Scale 7.

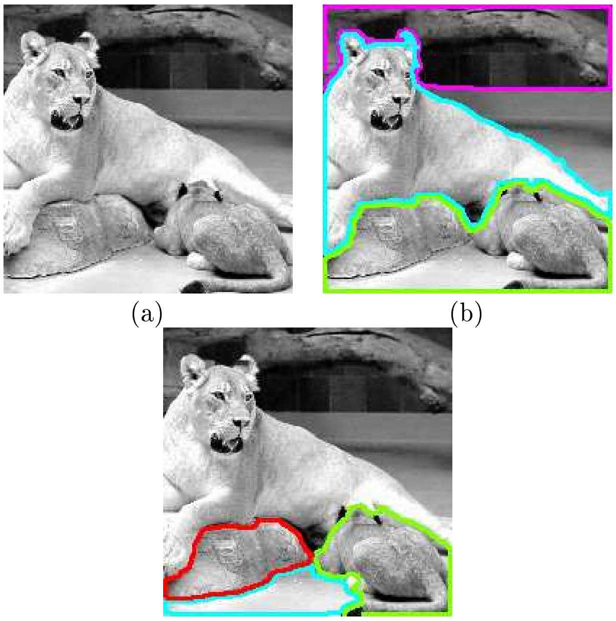
a single segment that includes the cow and the hilly background. At scale 9 the cow was separated from the hills, and the grass was split into two segments. Finally, in Figure 4 at the coarsest scale the grass was separated from the cows (except for the bright back of the cow which was decomposed later from the grass). The three cows were then split (with the rightmost cow split into two segments). Body parts of the cows are obtained in the lower scale. Overall, these pictures demonstrate that our algorithm accurately finds the relevant regions in the images.

7 Conclusion

We have introduced a fast, multiscale algorithm for image segmentation. The algorithm uses a process of recursive weighted aggregation to detect the distinctive segments at different scales. It finds an approximate solution to normalized cuts measure in time that is linear in the size of the image with only a few dozen operations per pixel. Future research directions include the use of various statistics to obtain segmentation based on richer information, improving the isotropy of the interpolations to produce smoother boundaries of segments, and combining the segmentation process with curve completion algorithms and top-down analysis of the image.

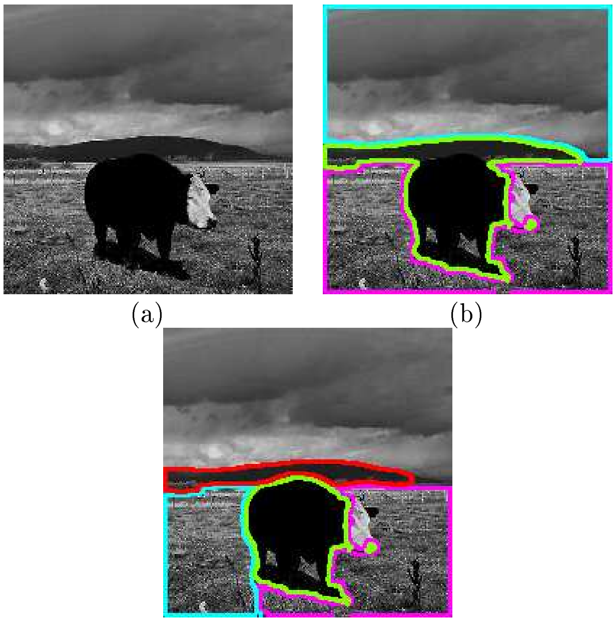
Appendix

The interpolation weights (14) can be improved, yielding a better approximation of the fine level minimization problem by the coarser representations and allowing us to represent the coarser problems with fewer block pixels. Ideally, the interpolation rule (5) should yield a fine-level configuration u that satisfies



(a) (b) (c)

Figure 2: (a) The input image. (b) Scale 10. (c) Scale 8.



(a) (b) (c)

Figure 3: (a) The input image. (b) Scale 10. (c) Scale 9.

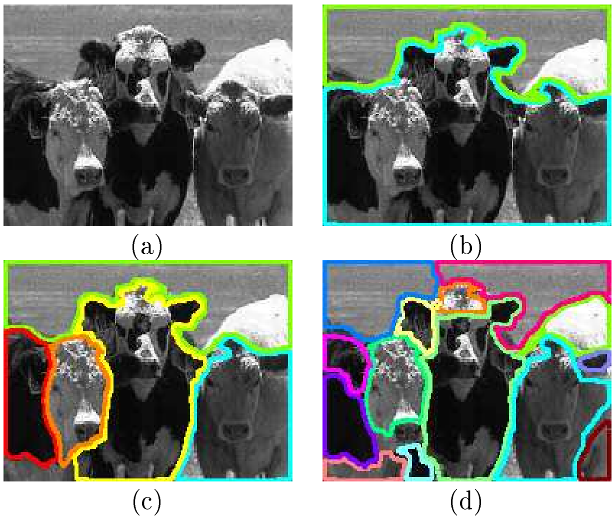


Figure 4: (a) The input image. (b) Scale 10. (c) Scale 9. (d) Scale 8.

the energy-minimization condition $\partial E(u)/\partial u_i = 0$. Since E is quadratic in u this condition can be written as

$$u_i^{(m)} = \sum_{j \in C} \hat{a}_{ij} u_j^{(m)} + \sum_{j \notin C} \hat{a}_{ij} u_j^{(m)} \;, \qquad (17)$$

where $\hat{a}_{ij}$ are the *normalized* couplings, defined by $\hat{a}_{ij} = a_{ij}/(\sum_l a_{il})$. Notice that the interpolation rule (5) considers only the first terms in (17). Given any (*non*-ideal) interpolation weights $\{w_{ik}\}$, *improved* interpolation weights $\{\bar{w}_{ik}\}$ are given by

$$\bar{w}_{ik} = \hat{a}_{ic_k} + \sum_{j \notin C} \hat{a}_{ij} w_{jk} \;. \qquad (18)$$

This same rule can recursively be reused several time, to create increasingly improved interpolation weights.

A measure of the "deficiency" d_i of interpolating to pixel i with the interpolation weights (14) is defined as the relative part of (17) being ignored by the relation (14), i.e., $d_i = \sum_{j \notin C} \hat{a}_{ij}$. Similarly, given any interpolation weights $\{w_{ik}\}$ with deficiencies $\{d_i\}$, the *improved* interpolation weights $\{\bar{w}_{ik}\}$ created by (18) will have the deficiencies $\bar{d}_i = \sum_{j \notin C} \hat{a}_{ij} d_j$. Hence, with reasonably dense set C, the deficiencies will be much reduced with each improvement, so that normally very few such improvements (if at all) would be needed. (Such improved interpolation rules are widely used in AMG, see, e.g., [2].)

With the improved interpolation weights (18), the coarse-variable selection criterion (11) can be relaxed, replacing it by the more general criterion $d_i \leq 1 - \beta$. Condition (16) can similarly be relaxed.

Finally, for computational efficiency it is desired to keep the interpolation matrix $\{w_{ik}\}$ as *sparse* (containing as few non-zero terms) as possible. For this purpose we replace small weights ($w_{ik} < \epsilon$, ϵ being another algorithm-control parameter; e.g., $\epsilon = .01$) by zeros, and then *renormalize* to maintain $\sum_k w_{ik} = 1$.

References

[1] S. Baronti, A. Casini, F. Lotti, L. Favaro, and V. Roberto, "Variable pyramid structure for image segmentation," *Comput. Vision Graphics Image Process.*, **49**: 346–356, 1990.

[2] A. Brandt, S. McCormick and J. Ruge, "Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations," Inst. for Computational Studies, POB 1852, Fort Colins, Colorado, 1982.

[3] A. Brandt, "Rigorous quantitative analysis of multigrid, I: Constant coefficients two-level cycles with L_2-norm," *SIAM J. Numer. Anal.*, **31**: 1695–1730, 1994.

[4] R. L. Cannon, R. L. Dave and J. C. Bezdek, "Efficient implementation of fuzzy c-means clustering algorithms," *IEEE Trans. Pattern Analysis and Machine Intellegence*, **8**(2), 1986.

[5] K. Cho and P. Meer, "Image segmentation from consensus information," *Computer Vision and Image Understanding*, **68**(1): 72–89, 1997.

[6] I. J. Cox, S. B. Rao and Y. Zhong, "Ratio Regions: A Technique for Image Segmentation," *Proc. Int. Conf. on Pattern Recognition*, **B**:557–564, August 1996.

[7] R. O. Duda and P. E. Hart, *Pattern classification and scene analysis.* Wiley-Interscience Publication, John Wiley and Sons, Inc, 1973.

[8] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **6**(6): 721–741, 1984.

[9] R.M. Haralick and L.G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Reading MA, 1992.

[10] J. M. Jolion and A. Rosenfeld, *A pyramid framework for early vision.* Kluwer Academic Publishers.

[11] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, **26**: 1277–1294, 1993.

[12] T. Pavlidis and Y. Liow, "Integrating region growing and edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**:255–233, 1990.

[13] P. Perona and W. Freeman, "A factorization approach to grouping," *Proceedings of European Conference on Computer Vision*: 655-670, Freiburg, Germany, 1998.

[14] S. Rao, "Faster algorithms for finding small separators in planar graphs," *Proc. of the ACM*, **15**(5):335–345, 1971.

[15] B. M. ter haar Romeny, ed., "Geometry-Driven Diffusion in Computer Vision," Kluwer Academic Pubs., 1994.

[16] S. Sarkar, "Learning to Form Large Groups of Salient Image Features," *Proc. of Computer Vision and Pattern Recognition*:780–786, 1998.

[17] J. Shi, J. Malik, "Normalized Cuts and Image Segmentation," *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*: 731–737, Puerto Rico, 1997.

[18] M. Spann and C. Horne, "Image segmentation using a dynamic thresholding pyramid," *Pattern Recognition*, **22**: 719–732, 1989.

[19] Y. Weiss, "Segmentation using eigenvectors: a unifying view," *International Conference on Computer Vision*: 975–982, 1999.

[20] Z. Wu and R. Leahy, "An optimal graph theoretic approach to data clustering: theory and its application to image segmentation," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **15**:1101–1113, 1993.