Segmentation of Sub-cortical Structures by the Graph-Shifts Algorithm

Jason J. Corso[1], Zhuowen Tu[1], Alan Yuille[2], and Arthur Toga[1]

[1] Center for Computational Biology
Laboratory of Neuro Imaging
[2] Department of Statistics
University of California, Los Angeles, USA
jcorso@ucla.edu

Abstract. We propose a novel algorithm called graph-shifts for performing image segmentation and labeling. This algorithm makes use of a dynamic hierarchical representation of the image. This representation allows each iteration of the algorithm to make both small and large changes in the segmentation, similar to PDE and split-and-merge methods, respectively. In particular, at each iteration we are able to rapidly compute and select the optimal change to be performed. We apply graph-shifts to the task of segmenting sub-cortical brain structures. First we formalize this task as energy function minimization where the energy terms are learned from a training set of labeled images. Then we apply the graph-shifts algorithm. We show that the labeling results are comparable in quantitative accuracy to other approaches but are obtained considerably faster: by orders of magnitude (roughly one minute). We also quantitatively demonstrate robustness to initialization and avoidance of local minima in which conventional boundary PDE methods fall.

1 Introduction

Segmenting an image into a number of labeled regions is a classic vision and medical imaging problem, see [1,2,3,4,5,6] for an introduction to the enormous literature. The problem is typically formulated in terms of minimizing an energy function or, equivalently, maximizing a posterior probability distribution. In this paper, we deal with a special case where the number of labels is fixed. Our specific application is to segment the sub-cortical structures of the brain, see section (2). The contribution of this paper is to provide a novel algorithm called *graph-shifts* which is extremely fast and effective for sub-cortical segmentation.

A variety of algorithms, reviewed in section (2), have been proposed to solve the energy minimization task for segmentation and labeling. For most of these algorithms, each iteration is restricted to small changes in the segmentation. For those methods which allow large changes, there is no procedure for rapidly calculating and selecting the change that most decreases the energy.

Graph-shifts is a novel algorithm that builds a dynamic hierarchical representation of the image. This representation enables the algorithm to make large

N. Karssemeijer and B. Lelieveldt (Eds.): IPMI 2007, LNCS 4584, pp. 183–197, 2007.

changes in the segmentation which can be thought of as a combined split and merge (see [4] for recent work on split and merge). Moreover, the graph-shifts algorithm is able to exploit the hierarchy to rapidly calculate and select the best change to make at each iteration. This gives an extremely fast algorithm which also has the ability to avoid local minima that might trap algorithms which rely on small local changes to the segmentation.

The hierarchy is structured as a set of nodes at a series of layers, see figure (1). The nodes at the bottom layer form the image lattice. Each node is constrained to have a single parent. All nodes are assigned a model label which is required to be the same as its parent's label. There is a neighborhood structure defined at all layers of the graph. A *graph shift* alters the hierarchy by changing the parent of a node, which alters the model label of the node and of all its descendants. This is illustrated in figure (1), which shows three steps in a three layer graph coloring potential shifts that would change the energy black and others gray. The algorithm can be understood intuitively in terms of competing crime families as portrayed in films like the Godfather. There is a hierarchical organization where each node owes allegiance to its unique parent node (or boss) and, in turn, to its boss's boss. This gives families of nodes which share the same allegiance (i.e. have the same model label). Each node has a subfamily of descendants. The top level nodes are the "bosses of all bosses" of the families. The graph-shifts algorithm proceeds by selecting a node to switch allegiance (i.e. model label) to the boss of a neighboring node. This causes the subfamily of the node to also switch allegiance. The algorithm minimizes a global energy and at each iteration selects the change of allegiance that maximally decreases the energy.

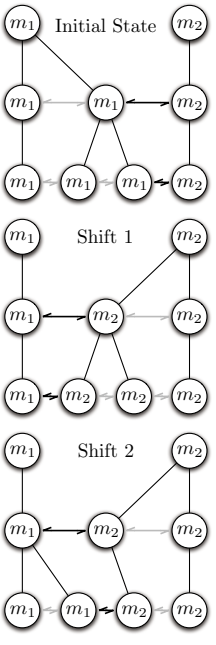


Fig. 1. Intuitive Graph-Shifts Example

The structure of this paper is as follows. In section (2) we give a brief background on segmentation. Section (3) describes the graph-shifts algorithm for a general class of segmentation problems. In section (4), we formulate the task of sub-cortical labeling in terms of energy function minimization and derive a graph-shifts algorithm. Section (5) gives experimental results and comparisons to other approaches.

2 Background

Many algorithms have been applied to segmentation, so we restrict our review to those methods most related to this paper. A common approach includes taking local gradients of the energy function at the region boundaries and thereby moving the boundaries. This region competition approach [2] can be successful

when used with good initialization, but its local nature means that at each iteration step it can only make small changes to the segmentation. This can cause slowness and also risks getting stuck in local minima. See [7] for similar types of partial differential equations (PDE) algorithms using level sets and related methods. Graph cuts [3] is an alternative deterministic energy minimization algorithm that can take large leaps in the energy space, but it can only be applied to a restricted class of energy functions (and is only guaranteed to converge for a subset of these) [8]. Simulated annealing [1] can in theory converge to the optimal solution of any energy function but, in practice, is extremely slow. The data-driven Markov chain Monte Carlo method [4] can combine classic methods, including split and merge, to make large changes in the segmentation at each iteration, but remains comparatively slow.

There have been surprisingly few attempts to define segmentation algorithms based on dynamic hierarchical representations. But we are influenced by two recent papers. Segmentation by Weighted Aggregation (SWA) [9] is a remarkably fast algorithm that builds a hierarchical representation of an image, but does not attempt to minimize a global energy function. Instead it outputs a hierarchy of segments which satisfy certain homogeneity properties. Moreover, its hierarchy is fixed and not dynamic. The multiscale Swendson-Wang algorithm [10] does attempt to provide samples from a global probability distribution. But it has only limited hierarchy dynamics and its convergence rate is comparatively slow compared to SWA. A third related hierarchical segmentation approach is proposed in [11], where a *hyperstack*, a Gaussian scale-space representation of the image, is used to perform a probabilistic linking (similar to region growing) of voxels and partial volumes in the scale-space. Finally, Tu [12] proposed a related segmentation algorithm that was similarly capable of making both small-scale boundary adjustments and large-scale split-merge moves. In his approach, however, a fixed size hierarchy is used, and the split-merge moves are attempted by a stochastic algorithm, which requires the evaluation of (often difficult to compute) proposal distributions.

Our application is the important task of sub-cortical segmentation from three-dimensional medical images. Recent work on this task includes [5,6,13,14,15,16]. These approaches typically formulate the task in terms of probabilistic estimation or, equivalently, energy function minimization. The approaches differ by the form of the energy function that they use and the algorithm chosen to minimize it. The algorithms are usually similar to those described above and suffer similar limitations in terms of convergence rates. In this paper, we will use a comparatively simple energy function similar to conditional random fields [17], where the energy terms are learned from training examples by the probabilistic boosting tree (PBT) learning algorithm [18].

3 Graph-Shifts

This section describes the basic ideas of the graph-shifts algorithm. We first describe the class of energy models that it can be applied to in section (3.1). Next

we describe the hierarchy in section (3.2), show how the energy can be computed recursively in section (3.3), and specify the general graph-shifts algorithm in section (3.4).

3.1 The Energy Models

The input image $\mathbf{I}$ is defined on a lattice D of pixels/voxels. For medical image applications this is a three-dimensional lattice. The lattice has the standard neighborhood structure and we define the notation $N(\mu, \nu) = 1$ if $\mu \in D$ and $\nu \in D$ are neighbors on the lattice, and $N(\mu, \nu) = 0$ otherwise. The task is to assign each voxel $\mu \in D$ to one of a fixed set of K models $m_\mu \in \{1, ..., K\}$. This assignment corresponds to a segmentation of the image into K, or more, connected regions.

We want the segmentation to minimize an energy function criterion:

$$E[\{m_\omega : \omega \in D\}] = \sum_{\nu \in D} E_1(\phi(\mathbf{I})(\nu), m_\nu) + \frac{1}{2} \sum_{\substack{\nu \in D, \mu \in D: \\ N(\nu,\mu)=1}} E_2(\mathbf{I}(\nu), \mathbf{I}(\mu), m_\nu, m_\mu). \tag{1}$$

In this paper, the second term E_2 is chosen to be a boundary term that pays a penalty only for neighboring pixels/voxels which have different model labels (i.e. $E_2(\mathbf{I}(\nu), \mathbf{I}(\mu), m_\nu, m_\mu) = 0$ if $m_\nu = m_\mu$). This penalty can either be a penalty for the length of the boundary, or may include a measure of the strength of local edge cues. It includes discretized versions of standard segmentation criteria such as boundary length $\int_{\delta R} ds$ and edge strength along boundary $\int_{\delta R} |\nabla \mathbf{I}|^2 ds$. (Here s denotes arc length, R denotes the regions with constant labels, and δR is their boundaries).

The first term E_1 gives local evidence that the pixel μ takes model m_μ, where $\phi(\mathbf{I})(\mu)$ denotes a nonlinear filter of the image evaluated at μ. In this paper, the nonlinear filter will give local context information and will be learned from training samples, as described in section (4.1). The model given in equation (1) includes a large class of existing models. It is restricted, however, by the requirement that the number of models is fixed and that the models have no unknown parameters.

3.2 The Hierarchy

We define a graph G to be a set of nodes $\mu \in \mathcal{U}$ and a set of edges. The graph is hierarchical and composed of multiple layers. The nodes at the lowest layer are the elements of the lattice D and the edges are defined to link neighbors on the lattice. The coarser layers are computed recursively, as will be described in section (4.2). Two nodes at a coarse layer are joined by an edge if any of their children are joined by an edge.

The nodes are constrained to have a single parent (except for the nodes at the top layer which have no parent) and every node has at least one child (except for nodes at the bottom layer). We use the notation $C(\mu)$ for the children of μ, and $A(\mu)$ for the parent. A node μ on the bottom layer (i.e. on the lattice) has

no children, and hence $C(\mu) = \emptyset$. We use the notation $N(\mu, \nu) = 1$ to indicate that nodes μ, ν on the same layer are neighbors, with $N(\mu, \nu) = 0$ otherwise.

At the top of the hierarchy, we define a special *root* layer of nodes comprised of a single node for each of the K model labels. We write $\overline{\mu}_k$ for these root nodes and use the notation $m_{\overline{\mu}_k}$ to denote the model variable associated with it. Each node is assigned a label that is constrained to be the label of its parent. Since, by construction, all non-root nodes can trace their ancestry back to a single root node, an instance of the graph G is equivalent to a labeled segmentation $\{m_\mu : \mu \in D\}$ of the image, see equation (1).

3.3 Recursive Computation of Energy

This section shows that we can decompose the energy into terms that can be assigned to each node of the hierarchy and computed recursively. This will be exploited in section (3.4) to enable us to rapidly compute the changes in energy caused by different graph shifts.

The energy function consists of regional and edge parts. These depend on the node descendants and, for the edge part, on the descendants of the neighbors. The regional energy E_1 for assigning a model m_μ to a node μ is defined recursively by:

$$E_1(\mu, m_\mu) = \begin{cases} E_1\left(\phi(\mathbf{I})(\mu), m_\mu\right) & \text{if } C(\mu) = \emptyset \\ \sum\limits_{\nu \in C(\mu)} E_1(\nu, m_\mu) & \text{otherwise} \end{cases} \tag{2}$$

where $E_1\left(\phi(\mathbf{I})(\mu), m_\mu\right)$ is the energy at the voxel from equation (1). The edge energy E_2 between nodes μ_1 and μ_2, with models m_{μ_1} and m_{μ_2} is defined recursively by:

$$E_2(\mu_1, \mu_2, m_{\mu_1}, m_{\mu_2}) = \begin{cases} E_2(\mathbf{I}(\mu_1), \mathbf{I}(\mu_2), m_{\mu_1}, m_{\mu_2}) & \text{if } C(\mu_1) = C(\mu_2) = \emptyset \\ \sum\limits_{\substack{\nu_1 \in C(\mu_1),\ \nu_2 \in C(\mu_2)\ : \\ N(\nu_1, \nu_2) = 1}} E_2(\nu_1, \nu_2, m_{\mu_1}, m_{\mu_2}) & \text{otherwise} \end{cases} \tag{3}$$

where $E_2(\mathbf{I}(\mu_1), \mathbf{I}(\mu_2), m_{\mu_1}, m_{\mu_2})$ is the edge energy for pixels/voxels in equation (1).

The overall energy (1) was specified at the voxel layer, but it can be computed at any layer of the hierarchy. For example, it can be computed at the top layer by:

$$E\left(\{m_{\overline{\mu}_k} : k = 1, ..., K\}\right) = \sum_{k=1}^{K} E_1(\overline{\mu}_k, m_{\overline{\mu}_k}) + \frac{1}{2} \sum_{\substack{i,j:1,..,K \\ N(\overline{\mu}_i, \overline{\mu}_j) = 1}} E_2(\overline{\mu}_i, \overline{\mu}_j, m_{\overline{\mu}_i}, m_{\overline{\mu}_j}). \tag{4}$$

3.4 Graph-Shifts

The basic idea of the graph-shifts algorithm is to allow a node μ to change its parent to the parent $A(\nu)$ of a neighboring node ν, as shown in figure (1). We will represent this shift as $\mu \rightarrow \nu$.

This shift not have any effect on the labeling of nodes unless the new parent has a different label than the old one (i.e. when $m_{A(\mu)} \neq m_{A(\nu)}$, or equivalently, $m_\mu \neq m_\nu$). In this case, the change in parents will cause the node and its descendants to change their labels to that of the new parent. This will alter the labeling of the nodes on the image lattice and hence will change the energy.

Consequently, we only need consider shifts between neighbors which have different labels. We can compute the changes in energy, or *shift-gradient* caused by these shifts by using the energy functions assigned to the nodes, as described in section (3.3). For example, the shift from μ to ν corresponds to a shift-gradient $\Delta E(\mu \to \nu)$:

$$\Delta E(\mu \to \nu) = E_1(\mu, m_\nu) - E_1(\mu, m_\mu) \; + \sum_{\eta: N(\mu,\eta)=1} [E_2(\mu, \eta, m_\nu, m_\eta) - E_2(\mu, \eta, m_\mu, m_\eta)] \quad . \tag{5}$$

The graph-shifts algorithm begins by initializing the graph hierarchy (section 4.2). Then we calculate the shift-gradients of all the shifts using equations (2),(3), and (5). We exploit recursion to calculate these shift-gradients extremely rapidly, see section (4.3). In practice, very few of the neighbors in the hierarchy have different labels and so the shift-gradients only need be computed for a small fraction of the total nodes. We throw away all shift-gradients which are positive or zero, since these shifts do not decrease the energy. The remaining shift-gradients are stored in a sorted, or unsorted, *shift-gradient list*, denoted S in figure 2 (we discuss the tradeoffs in section 4.3).

Graph-shifts proceeds by selecting the steepest shift-gradient in the list and makes the corresponding shift in the hierarchy. This changes the labels in the part of the hierarchy where the shift occurs, but leaves the remainder of the hierarchy unchanged. The algorithm recomputes the shift-gradients in the changed part of the hierarchy and updates the weight list. We repeat the process until convergence, when the shift-gradient list is empty (i.e. all shift-gradients in the graph are positive or zero).

GRAPH-SHIFTS
Input: Volume **I** on lattice D.
Output: Label volume **L** on lattice D.
0 Initialize graph hierarchy (figure 3).
1 Compute exhaustive set of potential shifts S.
2 while S is not empty
3 $s \leftarrow$ the shift in S that best reduces the energy.
4 Apply shift s to the graph.
5 Update affected region and edge properties.
6 Recompute affected shifts on boundary and update S. (5 & 6 discussed in section 4)
7 Compute label volume **L** from final hierarchy.

Fig. 2. Graph-shifts pseudo-code

Each shift is chosen to maximally decrease the energy, and so the algorithm is guaranteed to converge to, at least, a local minimum of the energy function. The algorithm prefers to select shifts at the coarser layers of the hierarchy, because these typically alter the labels of many nodes on the lattice and cause large changes in energy. These large changes can ensure that the algorithm can escape from some bad local minima.

4 Segmentation of 3D Medical Images

Now we describe the specific application to sub-cortical structures. The specific energy function is given in section (4.1). Sections (4.2) and (4.3) describe the initialization and how the shifts are computed and selected for the graph-shifts algorithm.

4.1 The Energy

Our implementation uses eight models for sub-cortical structures together with a background model for everything else. The regional terms $E_1(\mu, m_\mu)$ in the energy function (1) contain local evidence that a voxel μ is assigned a label m_μ. This local evidence will depend on a small region surrounding the voxel and hence is influenced by the local image context. We learn this local evidence from training data where the labeling is given by an expert.

We apply the probabilistic boosting tree (PBT) algorithm [18] to output a probability distribution $P(m_\mu|\phi(\mathbf{I})(\mu))$ for the label m_μ at voxel $\mu \in D$ conditioned on the response of a nonlinear filter $\phi(\mathbf{I})(\mu)$. This filter depends on voxels within an $11 \times 11 \times 11$ window centered on μ, and hence takes local image context into account. The non-linear filter ϕ is learned by the PBT algorithm which is an extension of the AdaBoost algorithm [19], [20]. PBT builds the filter $\phi(.)$ by combining a large number of elementary image features. These are selected from a set of 5,000 features which include Haar basis functions and histograms of the intensity gradient. The features are combined using weights which are also learned by the training algorithm.

We define the regional energy term by:

$$E_1(\mu, m_\mu) = -\log P(m_\mu|\phi(\mathbf{I})(\mu)), \tag{6}$$

which can be thought of as a pseudolikelihood approximation [18].

The edge energy term can take two forms. We can use it to either penalize the length of the segmentation boundaries, or to penalize the intensity edge strength along the segmentation boundaries. This gives two alternatives:

$$E_2(\mathbf{I}(\nu), \mathbf{I}(\mu), m_\nu, m_\mu) = 1 - \delta_{m_\nu, m_\mu}, \tag{7}$$

$$E_2(\mathbf{I}(\nu), \mathbf{I}(\mu), m_\nu, m_\mu) = \{1 - \delta_{m_\nu, m_\mu}\}\psi(\mathbf{I}(\mu), \mathbf{I}(\nu)), . \tag{8}$$

where $\psi(\mathbf{I}(\mu), \mathbf{I}(\nu))$ is a statistical likelihood measure of an edge between μ and ν; a simple example of such a measure is given in equation (9).

4.2 Initializing the Hierarchy

We propose a stochastic algorithm to quickly initialize the graph hierarchy that will be used during the graph shifts process. The algorithm recursively coarsens the graph by activating some edges according to the intensity gradient in the volume and grouping the resulting connected components up to a single node in the coarser graph layer. The coarsening procedure begins by defining a binary

edge activation variable $e_{\mu\nu}$ on each edge in the current graph layer G^t between neighboring nodes μ and ν (i.e., $N(\mu,\nu)=1$). The edge activation variables are then sampled according to

$$e_{\mu\nu} \sim \gamma\mathcal{U}(\{0,1\}) + (1-\gamma)\exp\left[-\alpha\left|\mathbf{I}(\mu)-\mathbf{I}(\nu)\right|\right] \tag{9}$$

where $\mathcal{U}$ is the uniform distribution on the binary set and γ is a relative weight between $\mathcal{U}$ and the conventional edge gradient affinity (right-hand side).

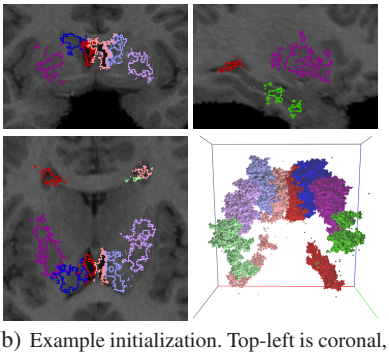
After the edge activation variables are sampled, a connected components algorithm is used to form node-groups based on the edge activation variables. The size of a connected component is constrained by a threshold τ, which governs the relative degree of coarsening between two graph layers. On the next graph layer, a node is created for each component. Following, edges in the new graph layer are induced by the connectivity on the current layer; i.e., two nodes in the coarse graph are connected if any two of their children are connected. The algorithm recursively executes this coarsening procedure until the size of the coarsened graph is within the range of the number of models, specified by a scalar β. Complete pseudo-code for this hierarchy initialization is given in figure 3.

Let G^T be the top layer of the graph hierarchy after initialization (example in figure 3(b)). Then, we append a model layer G^M on the hierarchy that contains a single node per model. Each node in G^T becomes the child of the node in G^M to which it has best fit, which is determined by evaluating the model fit $P(m|\mu)$ defined in section 4.1. One necessary constraint is that each node in G^M has at least one child in G^T, which is enforced by first linking each node in G^M to the node in G^T with highest probability to its model and the remaining links are created as described earlier.

HIERARCHY INITIALIZATION

Input: Volume $\mathbf{I}$ on lattice D.
Output: Graph hierarchy with layers $G^0,\ldots,G^T$.

```
Initialize graph G^0 from lattice D.
t ← 0.
repeat
  Sample edge activation variables in G^t using (9).
  Label every node in G^t as OPEN.
  while OPEN nodes remain in G^t.
    Create new, empty connected component C.
    Put a random OPEN node into queue Q.
    while Q is not empty and |C| < 1/τ.
      μ ← removed head of Q.
        Add ν to Q, s.t. N(μ,ν) = 1 and e_μν = 1.
        Add μ to C, label μ as CLOSED.
    Create G^{t+1} with a node for each C.
    Define I(C) as mean intensity of its children.
    Inherit connectivity in G^{t+1} from G^t.
    t ← t + 1.
until |G^t| < β * K.
```



(b) Example initialization. Top-left is coronal, top-right is sagittal, bottom-left is axial, and bottom-right is a 3D view.

Fig. 3. Initialization pseudo-code (left) and example (right)

4.3 Computing and Selecting Graph-Shifts

The efficiency of the graph-shifts algorithm relies on fast computation of potential shifts and fast selection of the optimal shift every iteration. We now describe how to satisfy these two requirements. To quickly compute potential shifts, we use an energy caching strategy that evaluates the recursive energy formulas (2) and (3) for the entire graph hierarchy following its creation (section 4.2). At each node, we evaluate and store the energies, denoted $\hat{E}_1(\mu, m_\mu) \doteq E_1(\mu, m_\mu)$ and $\hat{E}_2(\mu, \nu, m_\mu, m_\nu) \doteq E_2(\mu, \nu, m_\mu, m_\nu)$ for all μ, ν: $N(\mu, \nu) = 1$. These are quickly calculated in a recursive fashion. The computational cost of initializing the energy cache is $O(n \log n)$.

Subsequently, we apply the cached energy to evaluate the shift-gradient (5) in the entire hierarchy; this computation is $O(1)$ with the cache. At each node, we store the shift with the steepest gradient (largest negative ΔE), and discard any shift with non-negative gradient. The remaining shifts are stored in the *potential shift list*, denoted S in figure 2. In the volumes we have been studying, this list is quite small: typically only about 2% of all edges numbering about $60,000$ for volumes with 4 million voxels. The entire initialization including caching energies in the whole hierarchy takes $10 - 15$ seconds on these volumes, which amounts to about 30% of the total execution time.

At step **??** in the graph-shifts algorithm (figure 2), we must find the optimal shift in the potential shift list. One can use a sorted or unsorted list to store the potential shifts, with tradeoffs to both; an unsorted list requires no initial computation, no extra computation to add to the list, but an $O(n)$ search at each iteration to find the best shift. The sorted list carries an initial $O(n \log n)$ cost, an $O(\log n)$ cost for adding, but is $O(1)$ for selecting the optimal shift. Since every shift will cause modifications to the potential shift list, and the size of the list decreases with time (as fewer potential shifts exist), we choose to store an unsorted list and expend the linear search at each iteration.

As the graph shifts are applied, it is necessary to dynamically keep the hierarchy in synch with the energy landscape. Recomputing the entire energy cache and potential shift set is prohibitively expensive. Fortunately, it is not necessary: by construction, a shift is a very local change to the solution and only affects nodes along the boundary of the recently shifted subgraph. The number of affected nodes is dependent on the node connectivity and the height of the graph (it is $O(\log n)$); the node connectivity is relatively small and constant since the coarsening is roughly isotropic, and the height of the graph is logarithmic in the number of input voxels.

First, we update the energy cache associated with each affected node. This amounts to propagating the energy change up the graph to the roots. Let $\mu \rightarrow \nu$ be the executed shift. The region energy update must remove the energy contribution to $A(\mu)$ and add it to $A(\nu)$, which is the new parent of μ after the shift. The update rule is

$$\hat{E}_1(A(\mu), m_\mu))' = \hat{E}_1(A(\mu), m_\mu) - \hat{E}_1(\mu, m_\mu) \tag{10}$$

$$\hat{E}_1(A(\nu), m_\nu))' = \hat{E}_1(A(\nu), m_\nu) + \hat{E}_1(\mu, m_\nu) \ , \tag{11}$$

and it must be applied recursively to each parent until the root layer. Due to limited space, we do not discuss the details of the similar but more complicated procedure to update the edge energy cache terms $\hat{E}_2$. Both procedures are also $O(\log n)$.

Second, we update the potential shifts given the change in the hierarchy. All nodes along the shift boundary both below and above the shift layer must be updated because the change in the energy could result in changes to the shift-gradients, new potential shifts, and expired potential shifts (between two now nodes with the same model). Generally, this remains a small set since the shifts are local moves. As before, at each of these nodes, we compute and store the shift with the steepest negative gradient using the cached energies and discard any shift with a non-negative gradient or between two nodes with the same model. There are $O(\log n)$ affected shifts.

5 Experimental Results

A dataset of 28 high-resolution 3D SPGR T1-weighted MR images was acquired on a GE Signa 1.5T scanner as series of 124 contiguous 1.5 mm coronal slices

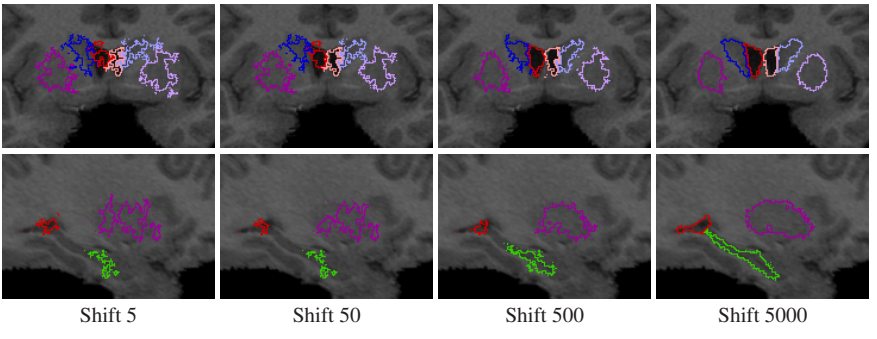


Fig. 4. Example of the graph-shifts process sampled during the minimization. Coronal and sagittal planes are shown, top and bottom respectively.

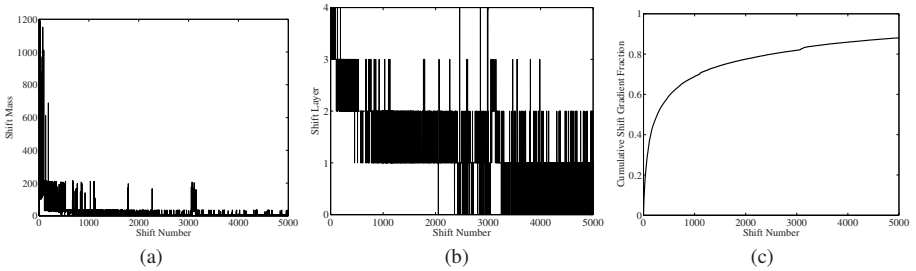


Fig. 5. (a) Graph shows the number of voxels (mass) that are moved per shift for 5000 shifts. (b) Graph shows the level in the hierarchy at which at shift occurs. (c) Graph shows the cumulative fraction of the total energy reduction that each shift effects.

(256x256 matrix; 20cm FOV). Brain volumes were roughly aligned and linearly scaled to perform 9 parameter registration. Four control points were manually given to perform this global registration. Expert neuro-anatomists manually labeled each volume into the following sub-cortical structures: hippocampus (LH, RH for left and right, resp. shown in green in figures), caudate (LC, RC in blue), putamen (LP, RP in purple), and ventricles (LV, RV, in red). We arbitrarily split the dataset in half and use 14 subjects for training and 14 for testing. The training volumes are used to learn the PBT region models and the boundary presence models. During the hierarchy initialization, we set the τ parameter to 0.15. We experimented with different values for τ, and found that varying it does not greatly affect the segmentation result.

The graph-shifts algorithm is very fast. We show an example process in figure 4 (this is the same volume as in figure 3(b)). Initialization, including the computation of the initial potential shift set, takes about 15 seconds. The remaining part of the graph shifts normally takes another 35 seconds to converge on a standard Linux PC workstation (2GB memory, 3Ghz cpu). Convergence occurs when no potential energy-reducing shift remains. Our speed is orders of magnitude faster than reported estimates on 3D medical segmentation: Yang et al. [13] is 120 minutes, FreeSurfer [5] is 30 minutes, Region Competition (PDE, obtained from a local implementation) is 5 minutes.

In figure 5-(c), we show the cumulative weight percentage of the same sequence of graph-shifts as figure 4. Here, we see that about 75% of the total energy reduction occurs within the first 1000 graph shifts. This large, early energy reduction corresponds to the shifts that occur at high layers in the hierarchy and have large masses as depicted in figure 5-(a) and (b). The mass of a shift is the number of voxels that are relabeled as a result of the operation. Yet, it is also clear from the plots that the graph-shifts at all levels at considered throughout the minimization process; recall, at any given time the potential shift list stores all energy reducing shifts and chooses the best one. Considering the majority of the energy reduction happens in the early stages of the graph-shift process, it is possible to stop the algorithm early when the shift gradient drops below a certain threshold.

In figure 6, we compare the total energy reduction of the dynamic hierarchical graph-shifts algorithm to the more conventional PDE-type energy minimization approach. To keep a fair comparison, we use the same structure and initial conditions in both cases. However, to approximate a PDE-type approach, we restrict the graph shifts to occur across single voxel boundaries (at the lowest layer in the hierarchy) only. As expected, the large-mass moves effect an exponential decrease in energy while the decrease from the single voxel moves is roughly linear.

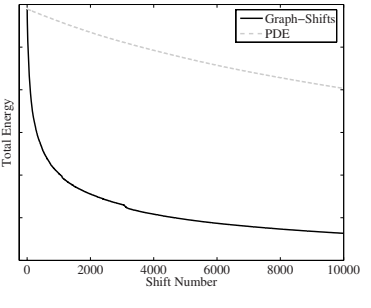


Fig. 6. Total energy reduction comparison of graph-shifts to a PDE method

Table 1. Segmentation accuracy using volume and surface measurements. A comparison to the FreeSurfer method run on the same data is included for the volume measurements in table 2.

	Training Set								Testing Set							
	LH	RH	LC	RC	LP	RP	LV	RV	LH	RH	LC	RC	LP	RP	LV	RV
Prec.	82%	70%	86%	86%	77%	81%	86%	86%	80%	58%	82%	84%	74%	74%	85%	85%
Rec.	60%	58%	82%	78%	72%	72%	88%	87%	61%	49%	81%	76%	67%	68%	87%	86%
Haus.	11.4	21.6	10.1	11.7	14.7	11.6	26.9	19.0	17.1	26.8	10.4	10.1	15.7	13.7	20.8	21.5
Mean	1.6	4.0	1.1	1.1	2.3	1.8	1.0	0.8	1.8	7.6	1.2	1.2	2.7	2.5	0.9	0.9
Med.	1.1	3.1	1.0	1.0	1.4	1.2	0.4	0.3	1.1	6.9	1.0	1.0	1.6	1.6	0.4	0.5

To quantify the accuracy of the segmentation, we use the standard volume (precision and recall), and surface distance (Hausdorff, mean and median) measurements. These are presented in table 1; in each case, the average over the set is given. In these experiments, we weighted the unary term four times as strong as the binary term; the power of the discriminative, context-sensitive models takes the majority of the energy while the binary term enforces local continuity and smoothness. Our accuracy is comparable or superior to the current state of the art in sub-cortical segmentation. To make a quantitative comparison, we computed the same scores using the FreeSurfer [5] method on the same data (results in table 2). We show a visual example of the segmentation in figure 7.

Table 2. FreeSurfer [5] accuracy

	LH	RH	LC	RC	LP	RP	LV	RV
Prec.	48%	51%	77%	78%	70%	76%	81%	69%
Rec.	67%	75%	78%	76%	83%	83%	76%	71%
Haus.	25.3	11.5	23.0	26.1	13.1	10.8	31.9	51.8
Mean	3.9	2.1	1.9	2.0	1.8	1.4	1.8	9.6
Med.	2.1	1.5	1.0	1.0	1.3	1.0	0.9	3.9

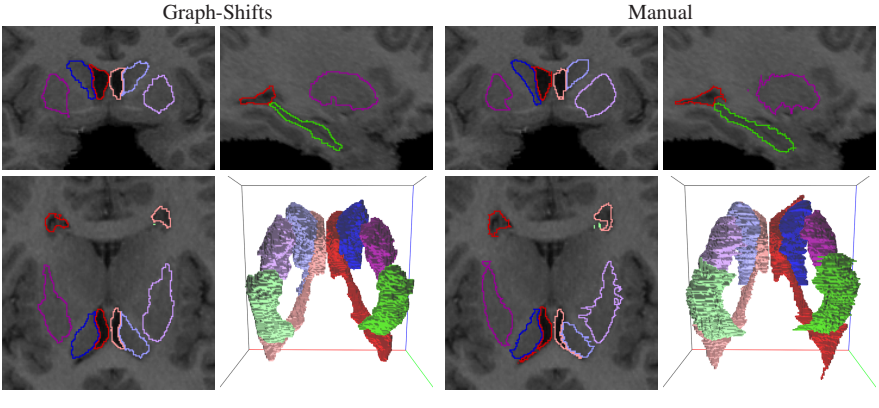


Fig. 7. An example of the sub-cortical structure segmentation result using the graph-shifts algorithm

Next, we show the graph-shifts algorithm is robust to initialization. We systematically perturbed the initial hierarchy by taking random shifts with positive gradient to increase the energy by 50%. Then, we started the graph-shifts from the degraded initial condition. In all cases, graph-shifts converged to (roughly) the same minimum; to quantify it, we calculated the standard deviation (SD) of the precision + recall score for over 100 instances. For all structures the SD is very small: LH: 0.0040, RH: 0.0011, LC: 0.0009, RC: 0.0013, LP: 0.0014, RP: 0.0013, LV: 0.0009, RV: 0.0014.

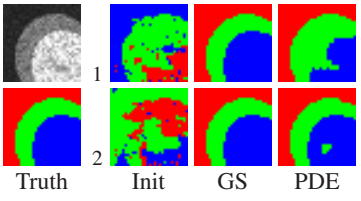


Fig. 8. Graph-shifts (GS) can avoid local minima. See text for details.

We now show that the large shifts provided by the hierarchical representation help avoid local minima in which PDE methods fall. We created a synthetic test image containing three separate i.i.d. Gaussian-distributed brightness models (depicted as red, green, and blue regions in figure 8). Following a similar perturbation as described above, we ran the graph-shifts algorithm as well as a PDE algorithm to compute the segmentation and reach a minimum. As expected, the graph-shifts method successfully avoids local minima that the PDE method falls into; in figure 8, we show two such cases. In the figure, the left column shows the input image and true labeling; the next three columns show the initial state, the graph-shifts result and the PDE result for two cases (rows 1 and 2).

6 Conclusion

We proposed graph-shifts, a novel energy minimization algorithm that manipulates a dynamic hierarchical decomposition of the image volume to rapidly and robustly minimize an energy function. We defined the class of energy functions it can minimize, and derived the recursive energy on the hierarchy. We discussed how the energy functions can include terms that are learned from labeled training data. The dynamic hierarchical representation makes it possible to make both large and small changes to the segmentation in a single operation, and the energy caching approach provides a deterministic way to rapidly compute and select the optimal move at each iteration.

We applied graph-shifts to the segmentation of sub-cortical brain structures in high-resolution MR 3D volumes. The quantified accuracy for both volume and surface distances is comparable or superior to the state-of-the-art for this problem, and the algorithm converges orders of magnitude faster than conventional minimization methods (about a minute). We demonstrated quantitative robustness to initialization and avoidance of local minima in which local boundary methods (e.g., PDE) fell.

In this paper, we considered the class of energies which used fixed model terms that were learned from training data. We are currently exploring extensions

to the graph-shifts algorithm that would update the model parameters during the minimization. To further improve sub-cortical segmentation, we are investigating a more sophisticated shape model as well as additional sub-cortical structures. Finally, we are conducting more comprehensive experiments using a larger dataset and cross-validation.

Acknowledgements

This work was funded by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 RR021813 entitled Center for Computational Biology (CCB). Information on the National Centers for Biomedical Computing can be obtained from `http://nihroadmap.nih.gov/bioinformatics`

References

1. Geman, S., Geman, D.: Stochastic Relaxation, Gibbs Distributions, and Bayesian Restoration of Images. IEEE Trans. on Pattern Analysis and Machine Intelligence 6, 721–741 (1984)
2. Zhu, S.C., Yuille, A.: Region Competition. IEEE Trans. on Pattern Analysis and Machine Intelligence 18(9), 884–900 (1996)
3. Boykov, Y., Veksler, O., Zabih, R.: Fast Approximate Energy Minimization via Graph Cuts. IEEE Trans. on Pattern Analysis and Machine Intelligence 23(11), 1222–1239 (2001)
4. Tu, Z., Zhu, S.C.: Image Segmentation by Data-Driven Markov Chain Monte Carlo. IEEE Trans. on Pattern Analysis and Machine Intelligence 24(5), 657–673 (2002)
5. Fischl, B., Salat, D.H., et al.: Whole brain segmentation: Automated labeling of neuroanatomical structures in the human brain. Neuron 33, 341–355 (2002)
6. Pohl, K.M., Fisher, J., Kikinis, R., Grimson, W.E.L., Wells, W.M.: A Bayesian Model for Joint Segmentation and Registration. NeuroImage 31, 228–239 (2006)
7. Chan, T.F., Shen, J.: Image Processing and Analysis: Variational, PDE, Wavelet, and Stochastic Methods. Soc. for Industrial and Applied Mathematics (SIAM), Phil., PA (2005)
8. Kolmogorov, V., Zabih, R.: What Energy Functions Can Be Minimized via Graph Cuts? In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 65–81. Springer, Heidelberg (2002)
9. Sharon, E., Brandt, A., Basri, R.: Fast Multiscale Image Segmentation. In: IEEE Conf. on Computer Vision and Pattern Recognition 2002, vol. I, pp. 70–77 (2000)
10. Barbu, A., Zhu, S.C.: Multigrid and Multi-level Swendsen-Wang Cuts for Hierarchic Graph Partitions. IEEE Computer Vision and Pattern Recognition, 731–738 (2004)
11. Vincken, K., Koster, A., Viergever, M.A.: Probabilistic Multiscale Image Segmentation. IEEE Trans. on Pattern Analysis and Machine Intelligence 19(2), 109–120 (1997)
12. Tu, Z.: An Integrated Framework for Image Segmentation and Perceptual Grouping. In: International Conference on Computer Vision 2005 (2005)
13. Yang, J., Staib, L.H., Duncan, J.S.: Neighbor-Constrained Segmentation with Level Set Based 3D Deformable Models. IEEE Trans. on Medical Imaging 23(8) (2004)

14. Pizer, S.M., Fletcher, P.T., et al.: Deformable m-reps for 3d medical image segmentation. International Journal of Computer Vision 55, 85–106 (2003)
15. Cocosco, C., Zijdenbos, A., Evans, A.: A Fully Automatic and Robust Brain MRI Tissue Classification Method. Medical Image Analysis 7, 513–527 (2003)
16. Wyatt, P.P., Noble, J.A.: MAP MRF joint segmentation and registration. In: Conference on Medical Image Computing and Computer-Assisted Intervention 2002, pp. 580–587 (2002)
17. Lafferty, J., McCallum, A., Pereira, F.: Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: Int. Conference on Machine Learning 2001 (2001)
18. Tu, Z.: Probabilistic Boosting-Tree: Learning Discriminative Models for Classification, Recognition, and Clustering. In: International Conference on Computer Vision 2005 (2005)
19. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-line Learning and an Application to Boosting. Journal of Computer and System Science 55(1), 119–139 (1997)
20. Schapire, R.E., Singer, Y.: Improved Boosting Algorithms Using Confidence-Rated Predictions. Machine Learning 37(3), 297–336 (1999)