Multi-Object Detection and Segmentation of Brain Structures Based on Dynamic Programming

Jue Wu and Albert C.S. Chung

Lo Kwee-Seong Medical Image Analysis Laboratory,
Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology, Hong Kong

Abstract. This work aims to design a detection and segmentation method using a graphical model in the context of multi-object brain image segmentation. We resort to dynamic programming as the optimization strategy to find the global minimum energy for the relation graph. Compared to other graphical models like tree structures, the proposed approach offers flexibility in accommodating more interactions among objects and thus can inhibit error propagation. Also, the new method is able to detect and segment a larger number of objects by searching for the global optimum energy in an efficient way. Experimental results show that the proposed approach achieves a comparable accuracy to other state-of-the-art methods.

1 Introduction

The goal of anatomical segmentation of human brains is to divide a brain image into anatomical subregions, such as, tissues and structures. Structure segmentation is more difficult than tissue extraction because a structure is usually a further segmentation of a tissue. However, structure segmentation plays a significant role in the study of human brain functioning and brain disease diagnosis [1–3]. Various methodologies have been proposed to tackle the difficult problem of brain structure segmentation [4–8]. Among others, tree-based methods [9–11] are emerging as an interesting way to manage the inter-relations between structures/objects. The weakness of a tree structure lies in the relatively simple interactions among nodes or objects. As a connected graph, a tree has the fewest edges, which equals the number of objects minus one. If more complicated interaction is desirable, then a tree structure is not satisfactory because no loop is allowed in a tree. On the other hand, the spatial relation among the brain structures is complex. In some cases, three or more structures are gathered together so it is hard to establish a hierarchical relationship like a tree among these structures.

In this paper, we present a novel dynamic programming-based graphical model to deal with brain structure segmentation. Dynamic programming (DP) is able to downsize the relation graph to a simple enough graph. As such, the relation graph can be complicated with many edges as long as DP is applicable. This overcomes the shortcoming of a tree graph with very limited number of edges. Furthermore, the DP-based method excels if the number of objects grows significantly. In this scenario, the tree-based methods tend to propagate errors along a single tree path due to the limited interactions. Note that there is only one path between any two nodes in a tree. Another important advantage of the dynamic programming-based method is the ability to find the global optimum in an efficient way. During the downsizing of the graph, DP keeps the records of the best candidates of the eliminated nodes and incorporates the associated energy in the subgraph. When the subgraph is optimized, the original graph is also optimized. The computational complexity of DP is exponentially lowered compared to an exhaustive search.

2 Methodology

2.1 Multi-Object Template Construction

Without prior information of shape and position, the task of extracting a pool of brain structures is very difficult since the structures often do not have clear boundaries in the brain. The template or atlas of multiple structures in the brain is served as the prior information for the detection and segmentation. It contains two-fold information. First, what are the shapes of the target structures? Second, what are the relations of position among these structures?

A legitimate template can be constructed from the ground truths of a set of segmentations. Multiple descriptions of the shapes and relative positions of the structures can be combined by a certain strategy. In this paper, we adopt a quick and simple strategy to construct the template so we use the ground truth of one segmentation. The ground truth delimitates the shape of each structure and determines position relations among the structures. We assume that different subjects have similar profiles in the two aspects (shape and position).

We create a graph to further define the relations among the structures. The vertices in the graph represent objects/structures. If two objects interact, there is an edge connecting the corresponding vertices. The weight of a vertex is pertaining to the similarity between the object and the superimposed image. The weight of an edge reflects the relative position between two interactive objects. The aim is to minimize the total weights associated with the whole graph. We create a graph that is suitable for dynamic programming to apply. The requirement is that the graph can be reduced into a smaller graph with fewer vertices. When the original graph is optimized, the smaller graph also reaches an optimum. We call such a graph a DP-graph. The vertices in a DP-graph can be eliminated one by one on a certain order until a manageable size of vertices is remained. After one vertex is eliminated, all the weights associated to it are incorporated to the smaller graph through a bookkeeping of the best candidates.

Theoretically, dynamic programming can be applied to many kinds of graphs. In the current context, we apply it to a triangulated graph which strikes a good balance between complexity and efficiency [12]. We target to detect and segment 12 structures in the brain and the constructed graph has 15 vertices. Three structures (lateral ventricle, the third ventricle and the fourth ventricle) are segmented in advance and serve as the base objects. The other objects are left/right thalamus, caudate, putamen, pallidum, hippocampus and amygdala. The interactions among them are shown in Fig. 1. We build these edges according to the spatial proximity of the objects.

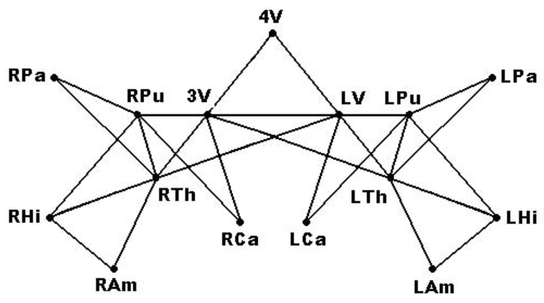

Fig. 1. The relation graph for 15 brain structures. LV: lateral ventricle, 3V: the third ventricle, 4V: the fourth ventricle, L/RTh: left/right thalamus, L/RCa: left/right caudate, L/RPu: left/right putamen, L/RPa: left/right pallidum, L/RHi: left/right hippocampus, L/RAm: left/right amygdala.

2.2 Matching Measures

We formulate the matching process between a template and an image as an optimization process of a specific energy. Each objects in the template can be superimposed on the image in several candidate positions. As such, the multi-object template has many configurations based on the combination of different positions of each object. We design an energy to measure both the similarity between each object and the underlying image region and the proper distances among the objects. The energy is composed of two components. One is the single-object measure and the other is the measure for two interactive objects. The two measures supplement each other. The unary measure is used to draw one object to the position where a brain structure exists probably. The binary measure regulates the inter-position between two interactive objects. The total energy is expressed by,

$$E_T(P) = \alpha \sum_i E_u(p_i|I) + \sum_i \sum_{j \in N_i} E_b(p_i, p_j), \tag{1}$$

where $E_T(P)$ denotes the objective total energy of the model, E_u is a unary measurement for a single object, p_i is the position of part i represented by spatial coordinates, E_b measures a compatibility between two objects, P is the set of possible positions of all objects, I is the input image, and N_i is the set of objects linking to object i.

The unary measure assumes that (1) the intensity level inside one object is approximately homogeneous and its variation should be small; (2) The voxels around the boundary of the object usually undergo an abrupt change of intensity and the gradient magnitude therein should be large. The two assumptions are valid considering that the brain structures are usually made of the same tissue type, e.g. white matter or grey matter, and they are often surrounded by a different tissue although the boundary may be lowly contrasted. The premises lead to the following formulation for the unary measures:

$$E_u(p_i|I) = \frac{1}{a_i}\iiint\limits_{A_i}(I(x,y,z)-\overline{I}(A_i))^2dxdydz - \frac{\beta_i}{b_i}\iiint\limits_{B_i}|\nabla I(x,y,x)|dxdydz, \tag{2}$$

where $\overline{I}(A_i)$ is the mean intensity in the image region A_i that is overlapped with object i and a_i is the volume of the region A_i. B_i is the boundary of object i, b_i is the area of the boundary B_i and $I(x,y,z)$ is the intensity of the voxel at coordinate (x,y,z).

The binary measure adjusts the relative position of two objects that are connected by one edge in the graph. It is content free and does not depend on the image data. When two objects are too close or too faraway compared to the beginning status, the measure imposes a large penalty. If two objects have a significant overlapping, the unreal configuration is also punished by the measure. The binary measure is defined as,

$$E_b(p_i,p_j) = ||(C_{p_i}-C_{p_j})-(\hat{C}_{p_i}-\hat{C}_{p_j})|| + L\cdot I\{a_{ij}/a_j > \tau\}, \tag{3}$$

where C_{p_i} is the coordinate vector of the center of mass of object i at position p_i and $\hat{C}_{p_i}$ is the initial center of mass of object i. $||\boldsymbol{v}||$ denotes the norm of vector $\boldsymbol{v}$. $I\{\text{event}\}$ is an indicator function, i.e., if the event is true, $I=1$; otherwise $I=0$. a_j is the volume of object j, a_{ij} is the volume of the overlapping region between objects i and j, and τ is a tolerance factor with respect to the overlapping extent. L is a very large positive penalty.

The parameters α and β determine the weights between terms in the energy. If we allow loose interaction among objects, we can set α to a large value. β helps to reduce the difference in magnitude between the gradient and the variance. It can be set according to the ground truth of the training image as follows,

$$\beta_i = (\frac{1}{a_i^t}\iiint\limits_{A_i^t}(I(x,y,z)-\overline{I}(A_i^t))^2dxdydz)/(\frac{1}{b_i^t}\iiint\limits_{B_i^t}|\nabla I(x,y,x)|dxdydz), \tag{4}$$

where x^t represents the same quantity $x(x = b_i, A_i, a_i, B_i)$ as aforementioned except with the known position of object i superimposed on the training image.

2.3 Optimization by Dynamic Programming

Once the energy is defined, we can then move the template around the input image to find the optimal position for each object. In the process we regard that positions with smaller energy are better than those with larger energy. The optimization is challenging since the search space is huge. Suppose there are n objects in a triangulated graph and for each object there are k possible positions to choose. If we enforce exhaustive search for a global minimum, we need to calculate the energy k^n times. However, if we apply dynamic programming to the optimization, the energy is calculated only $(n-2)k^3$ times with increased spatial requirement.

Dynamic programming (DP) is a strategy that solves a problem by downsizing. A problem suitable for DP can usually be formulated in a recursive way. DP simplifies the problem by solving a less complicated sub-problem first. When the original problem is optimized, the sub-problem is also optimized on a smaller support. Dynamic programming does the bookkeeping and records the optimal extension from the sub-problem to the original problem. The downsizing continues until a sub-problem with a manageable scale is obtained and can be solved easily. Afterwards, the solution of the smallest sub-problem is used to track back to the solution of the original problem.

Dynamic programming can be applied to problems in various domains. In the present context, a graph is eligible if there is always a cell (a connected subgraph with 3 or more vertices) that has only one shared edge (we call it an anchor edge) with the rest part of the graph before each downsizing. All the edges except the anchor edge are exclusive to the cell. We call such a cell a dangling cell. The simplest dangling cell is a triangle. It is not necessary that the graph has only one kind of cell. The DP strategy is to collapse the dangling cell to the anchor edge. The best configurations in the dangling cell for every possible configuration of the two vertices of the anchor edge are memorized. Collapse the cells one by one and the original graph is downsized to a much smaller scale. In each collapse, the vertices in the dangling cell are eliminated except for the two vertices of the anchor edge.

Take the current setting shown in Fig. 1 as an example. The cells are all triangles. There are more than one dangling cells in the present iteration, e.g, LAm-LHi-LTh, LPa-LPu-LTh, LV-LCa-LPu, and their anchor edges are LTh-LHi, LPu-LTh and LV-LPu, respectively. We can choose any of them to start collapsing. The sequence of vertices to be eliminated is called the elimination order. We adopt for the graph an elimination order of (LAm, LHi, LPa, LCa, LPu, LTh, RAm, RHi, RPa, RCa, RPu, RTh). After the 12 vertices are eliminated, the graph is downsized to a single triangle which can be simply optimized by brute force. The bookkeeping makes it possible to transit from a large graph to a smaller one without loss of information. The bookkeeping is the process of memorizing the best position for an eliminated vertex given the positions of the vertices of the anchor edge in the same dangling cell. Note that all the k^2 position combinations of the vertices of the anchor edge need to be kept for later use. For example, during the elimination of vertex LAm, DP memorizes the best position for each position combination of LHi and LTh. The corresponding weights for each configuration are also registered, i.e. the sum of three weights on vertex LAm, on edge LAm-LHi and edge LAm-LTh. Then, replace the old weight on the anchor edge LHi-LTh with the above sum after deleting LAm and its edges. Thus a smaller subgraph is generated. The iteration continues until three vertices 4V, 3V and LV are left. We assume the true configuration of the smallest subgraph is easily obtained. The backtracking from the last triangle to the other ones is achievable thanks to the bookkeeping during downsizing. We now know the best choices of the positions of some initial vertices, e.g. 4V, 3V and LV. In the reverse order of elimination, we can find the best position of the other vertices. For instance, the optimal position of RTh can be retrieved since it is recorded in the bookkeeping of 3V and LV. 3V, LV and RTh are once in the same dangling cell during the elimination and recall that the best position of RTh is kept for every possible position combination of 3V and LV during the bookkeeping process. The backtracking is then repeated for RPu, RCa and so on. In the end, the optimal positions for all the eliminated vertices are found.

The pseudocode of the dynamic programming for the graph is shown in Algorithm 1. The notations are as follows. N is the number of objects in the graph. s is the number of possible poses for each object. $\texttt{Edge}(p, q, i)$ is the optimal energy accumulated on edge i when the two objects a and b linked by edge i have poses of p and q. $\texttt{Adj}(i, j)$ is the index (the 3rd dimension in array $\texttt{Edge}$) of the edge between objects i and j, if i and j are adjacent. $\texttt{Adj}(i, j) = 0$ otherwise. $\texttt{ElimOrder}(i)$ is the ith object to eliminate. $\texttt{Tri}(i, 1), \texttt{Tri}(i, 2)$ and $\texttt{Tri}(i, 3)$ are the three objects indices in triangle i. $\texttt{Tri}(i, 1)$ is the first to eliminate. $\texttt{Triangle}(p, q, i)$ is the optimal pose of the object $\texttt{Tri2Obj}(i)$ which is the first-to-eliminate object in triangle i and the two other objects a and b in triangle i have poses of p and q. $\texttt{Obj2Tri}(i)$ is the index (the 3rd dim in array $\texttt{Triangle}$) of the triangle which the first-to-eliminate object is i.

It is worth pointing out that though the current setting of triangular collapsing is similar to some existing methods [13, 14], the dynamic programming-based framework can be applied to more complicated situations since the dangling cell can be far more complex than a triangle.

After the dynamic programming is finished, we fine-tune the coarse segmentation by means of the non-rigid B-spline registration between the input image and the moved trained image [10]. The moved training image has been deformed according to the displaced template obtained from the DP process. The similarity metric for the B-spline registration is mutual information. The final segmentation is thus obtained by propagating the registered template label to the input image.

3 Experiment

In the experiments, we applied the proposed method to the public database, IBSR [15], which contains 18 sets of T1-weighted brain MR images with expert-segmented internal structures. The bias-corrected images are of $256 \times 256 \times 128$

Algorithm 1 : Dynamic programming for a triangulated graph

```
**Require:** ElimOrder, Edge, Tri, Triangle, Obj2Tri
**Ensure:** pos, optimal
  for each edge $v$ do
    for $s_1 = 1, \cdots, s$ do
      for $s_2 = 1, \cdots, s$ do
        Edge$(s_1, s_2, v)$ = the binary measure for one object in position $s_1$ and the
        other object in position $s_2$ (the two objects are linked by edge $v$);
      end for
    end for
  end for
  for $i = 1, \cdots, n-3$ do
    $c =$ ElimOrder$(i)$, $a =$ Tri(Obj2Tri$(c)$, 3), $b =$ Tri(Obj2Tri$(c)$, 2);
    for $p = 1, \cdots, s$ do
      for $q = 1, \cdots, s$ do
        minEnergy = a large number;
        for $r = 1, \cdots, s$ do
          $e$ = accumulated energy on the edge linking $a$ and $c$ + accumulated energy
          on the edge linking $b$ and $c$ + unary energy for object $c$ in position $r$;
          if $e <$ minEnergy then
            minEnergy $= e$;
            Triangle$(p, q,$ Obj2Tri$(c)) = r$;
          end if
        end for
        Increase Edge$(p, q,$ Adj$(a, b))$ by minEnergy;
      end for
    end for
  end for
  optimal $= \min_{p,q}$(Edge(p, q, 1));
  for $i = n-3, \cdots, 1$ do
    $c =$ ElimOrder(i);
    $t =$ Obj2Tri$(c)$;
    pos$(c)$ = Triangle(pos(Tri$(t, 2)$), pos(Tri$(t, 3)$, $t$);
  end for
  Return pos and optimal;
```

voxels with various voxel sizes. Some of the target structures in the data sets are of low contrast and hard to detect even for human visual inspection.

Before executing the algorithm, we set the parameters of the method once and they were fixed in all the tests. α was set to 4 since the average degree of the vertices in the graph is around 4. We believe the overlapping of more than 10% of the object region is unacceptable so τ was set to 0.1. The grid size of the third-order B-spline was $5 \times 5 \times 5$. The search space for object positions was set to be within the range of $5 \times 5 \times 3$ voxels from the initial positions and hence $k = 75$ poses were explored for each object/structure. The magnitude of the whole search space for 12 objects was 75^{12}. The segmentations of 4V, 3V and LV were acquired from the training. Alternatively, we can adopt an automatic method to segment the ventricles as in [10]. We adopted the commonly-used Dice metric to quantify the difference between the segmentation results and the ground truths. The Dice score is defined as Dice $= (2||A \bigcap B||)/(||A|| + ||B||)$, where A and B are two shapes to be compared. Dice equal to 1 means a perfect match between two shapes and Dice equal to zero means no overlapping. The larger the Dice score is, the more overlapping the two shapes have.

We randomly chose a subject in the data sets as the only template. Then the proposed method was tested on the other 17 data sets. The Dice scores for the segmentation results before and after the application of the proposed approach are shown in Fig. 2. Here the initial segmentations are the regions superimposed by the template before we apply the proposed method. It is observed that the proposed method improves the segmentation of all structures by more than 10 percentage points on average. When the initial segmentations have a large variance, the DP-based method can decrease the variance and obtains more consistent results. The experiments were run in MatLab codes on a 2.13 GH CPU with 1 GB memory. The average running time is around 74 minutes per data set. We also show one example of the original image, ground truth and the segmentation result in Fig. 3.

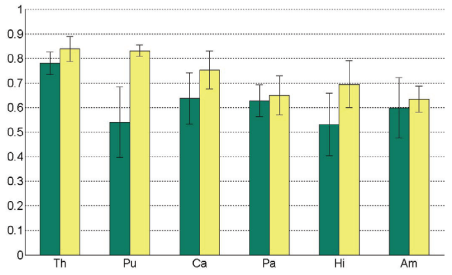

Fig. 2. The average Dice scores and their standard deviations for the initial segmentation (left green bars) and the final segmentation (right yellow bars). Th: thalamus, Pu: putamen, Ca: caudate, Pa: pallidum, Hi: hippocampus, Am: amygdala. Results of the same type of left and right structures are combined.

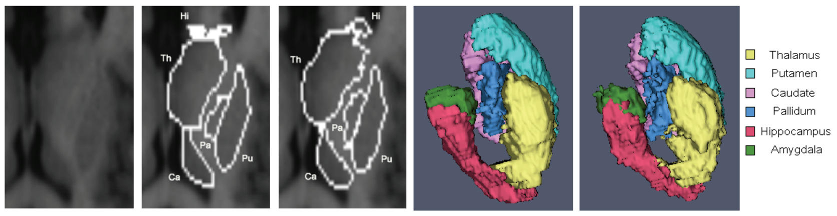

Fig. 3. One slice of a data set and corresponding segmentations. From left to right are original image, ground truth, segmentation result, 3D visualization of the ground truth and 3D visualization of the segmentation result.

4 Discussion and Summary

The same as other graphical models [10], the proposed method has some advantages over other methodologies, such as level sets, fuzzy logic and registration-based segmentation. For example, the graphical methods are usually more efficient than registration-based segmentation because the former processes the structures directly while the latter makes use of the whole brain information. Level set methods require the initialization to be close to the target. Otherwise, the level sets will converge to irrelevant structures in the brain. The proposed method circumvents the initialization problem by finding the global optimum through dynamic programming.

Quantitative comparison of the proposed method with the related methods is not conclusive because the methods were tested on different databases, targeted different sets of structures, or evaluated by different similarity metrics. The proposed method performed slightly better than a related method that was also tested on IBSR [16]. Their Dice scores for caudate, hippocampus, amygdala, putamen and pallidum were 0.76, 0.67, 0.63, 0.78 and 0.71. Ours are 0.76, 0.70, 0.64, 0.83 and 0.65. Generally speaking, in the related methods [5–7], the Dice scores for caudate, putamen and thalamus ranged from 0.75 to 0.90. The scores for pallidum, hippocampus and amygdala ranged from 0.60 to 0.75. The proposed method achieved comparable accuracies.

One limitation about the current method is that although the dynamic programming is capable of finding the global minimal energy, the energy formulation itself is not perfect. It is not ensured that the global minimal energy coincides with the true structure positions. It is an even more complicated case considering that the template possesses moderately different shapes from the true shapes. Theoretically, during the position search, we can allow sophisticated shape search simultaneously but this will make the computational burden formidable. It remains a profound future topic how to design a nearly-perfect energy to accommodate various shape changes while enabling efficient optimizations. All in all, the current work upgrades previous methods [10, 11] based on simple graphs (e.g. trees) to a more advanced method. This DP-based method allows more interaction in the relation graph and thus remains stable and robust to error propagation in spite of more objects.

References

1. Grenander, U., Miller, M.: Computational anatomy: An emerging discipline. Quarterly of Applied Mathematics **56** (1998) 617 – 694
2. Kandel, E., Schwartz, J., Jessell, T.: Principles of Neural Science. 4 edn. New York: McGraw-Hill (2000)
3. Packard, M., Knowlton, B.: Learning and memory functions of the basal ganglia. Annual Review Neuroscience **25** (2002) 563 – 593
4. Ciofolo, C., Barillot, C.: Brain segmentation with competitive level sets and fuzzy control. In: Information Processing in Medical Imaging. Volume LNCS 3565. (2005) 333 – 344
5. Zhou, J., Rajapakse, J.: Segmentation of subcortical brain structures using fuzzy templates. NeuroImage **28** (2005) 915 – 924
6. Khan, A., Wang, L., Beg, M.: Freesurfer-initiated fully-automated subcortical brain segmentation in mri using large deformation diffeomorphic metric mapping. NeuroImage **41** (2008) 735 – 746
7. Powell, S., Magnotta, V., Johnson, H., Jammalamadaka, V., Pierson, R., Andreasen, N.: Registration and machine learning-based automated segmentation of subcortical and cerebellar brain structures. NeuroImage **39** (2008) 238 – 247
8. Tu, Z., Narr, K., Dollar, P., Dinov, I., Thompson, P., Toga, A.: Brain anatomical structure segmentation by hybrid discriminative/generative models. IEEE Trans. on Medical Imaging **27**(4) (2008) 495 – 508
9. Pohl, K., Bouix, S., Nakamura, M., Rohlfing, T., McCarley, R., Kikinis, R., Grimson, W., Shenton, M., Wells, W.: A hierarchical algorithm for mr brain image parcellation. IEEE Trans. Medical Imaging **26** (2007) 1201 – 1212
10. Wu, J., Chung, A.: Markov dependence tree-based segmentation of deep brain structures. In: MICCAI. Volume LNCS 5242. (2008) 1092 1100
11. Wu, J., Chung, A.: A novel framework for segmentation of deep brain structures based markov dependence tree. NeuroImage **46** (2009) 1027 – 1036
12. Amit, Y.: 2D Object Detection and Recognition. The MIT Press (2002)
13. Felzenszwalb, P., Huttenlocher, D.: Efficient matching of pictorial structures. In: Proc. IEEE Computer Vision and Pattern Recognition Conf. (2000) 66–73
14. Felzenszwalb, P., Huttenlocher, D.: Representation and detection of deformable shapes. IEEE PAMI **27**(2) (2005) 208–220
15. Internet Brain Segmentation Repository: www.cma.mgh.harvard.edu/ibsr/.
16. Gouttard, S., Styner, M., Joshi, S., Smith, R., Cody, H., Gerig, G.: Subcortical structure segmentation using probabilistic atlas priors. In: SPIE Medical Imaging. Volume 6512. (2007) 65122j–11