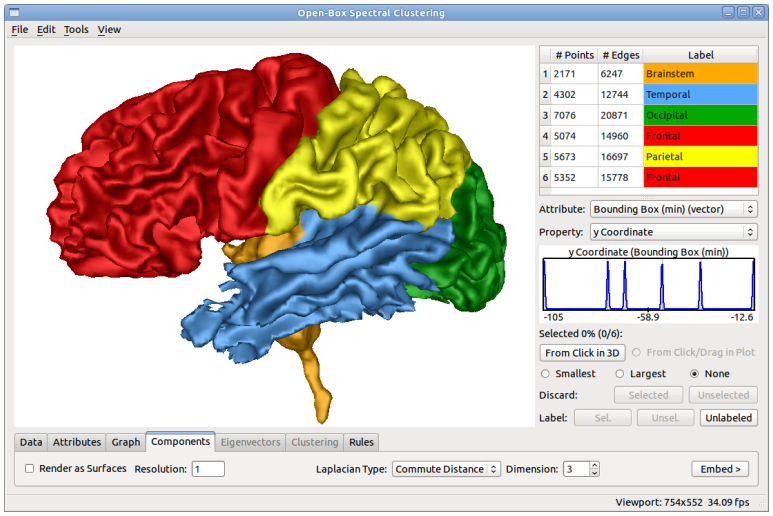
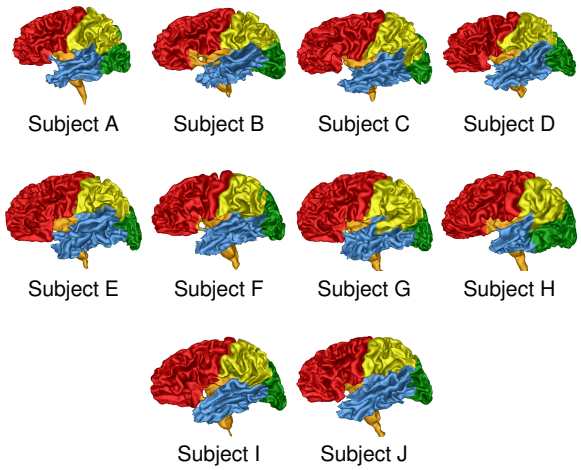
Open-Box Spectral Clustering: Applications to Medical Image Analysis

Thomas Schultz and Gordon L. Kindlmann



(a) Our system for interactive open-box spectral clustering.



(b) Results from transferring the user actions performed on the subject shown in (a) to ten other subjects.

Fig. 1. Adapting spectral clustering to specific image analysis problems involves tuning parameters and exploring hierarchical clustering strategies, different graph types and spectral embeddings. We present a Visual Analytics system that supports this process and finally outputs rules that can be successfully applied to similar data.

Abstract—Spectral clustering is a powerful and versatile technique, whose broad range of applications includes 3D image analysis. However, its practical use often involves a tedious and time-consuming process of tuning parameters and making application-specific choices. In the absence of training data with labeled clusters, help from a human analyst is required to decide the number of clusters, to determine whether hierarchical clustering is needed, and to define the appropriate distance measures, parameters of the underlying graph, and type of graph Laplacian. We propose to simplify this process via an open-box approach, in which an interactive system visualizes the involved mathematical quantities, suggests parameter values, and provides immediate feedback to support the required decisions. Our framework focuses on applications in 3D image analysis, and links the abstract high-dimensional feature space used in spectral clustering to the three-dimensional data space. This provides a better understanding of the technique, and helps the analyst predict how well specific parameter settings will generalize to similar tasks. In addition, our system supports filtering outliers and labeling the final clusters in such a way that user actions can be recorded and transferred to different data in which the same structures are to be found. Our system supports a wide range of inputs, including triangular meshes, regular grids, and point clouds. We use our system to develop segmentation protocols in chest CT and brain MRI that are then successfully applied to other datasets in an automated manner.

Index Terms—Image segmentation, spectral clustering, high-dimensional embeddings, linked views, programming with example

1 Introduction

Despite decades of active research, image segmentation remains a problem that lacks a fully automatic and generally applicable solution. This is not surprising, since the task in segmentation is to delineate meaningful image structures, and it is highly problem specific what constitutes a meaningful structure. Therefore, even though several powerful and flexible segmentation algorithms exist, they have to be customized to each individual task. This often involves a tedious and time-consuming process in which a human analyst formalizes the properties of the desired structure, develops a strategy for its automated recognition, and tunes parameters of the algorithm. Usually, this is done by iteratively modifying a piece of program code or its parameters, running it, and observing the changes in the results.

Even though it may appear natural to take a Visual Analytics approach to this problem, there exists very little work that supports this process with interactive visual techniques. The most closely related work we are aware of is Tuner [48], an interactive systems for exploring the parameter space of image segmentation algorithms in a systematic way. By keeping track of the settings for which the algorithm has already been run and predicting the segmentation quality that other settings might achieve, Tuner helps the user to identify a region in the parameter space in which the resulting segmentation has high quality and does not depend strongly on the exact parameter values.

Tuner's strategy is agnostic to the segmentation method ("black box" approach), which has the advantage that it can be combined

- *Thomas Schultz is with the University of Bonn, Germany. E-mail: schultz@cs.uni-bonn.de.*
- *Gordon Kindlmann is with the University of Chicago, USA. E-mail: glk@uchicago.edu.*

Manuscript received 31 March 2013; accepted 1 August 2013; posted online 13 October 2013; mailed on 4 October 2013.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

1077-2626/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

with different algorithms without requiring any changes to the system. However, it assumes that all relevant parameters are continuous quantities; that a quality measure is available to judge segmentation results, which Tuner obtains by comparing the achieved segmentation to a ground truth; that this quality depends continuously on the segmentation parameters; and that individual runs of the segmentation method are not too costly, since a large enough number of initial runs is needed to obtain a realistic impression of the parameter space on which further refinement can build.

When its assumptions are met, Tuner can greatly reduce the time spent on parameter tuning. However, we believe that if one or several of them are violated, it can be beneficial to instead take an open-box approach, in which visualization makes key elements of the segmentation method transparent to the user. In this case, the user's decisions are not only informed by the final outcome, but also by an improved understanding of how the algorithm arrives at that result.

This paper presents such an approach. By the nature of an open-box system, it has to specialize to one specific segmentation method. In this work, we select spectral clustering [51], which has proven powerful to segment not only images, but also graphs that describe more abstract data. This is particularly useful in our example application, three-dimensional medical image analysis, since in this context, segmentation is often not performed directly on the individual voxels, but on higher-level geometrical features, such as isosurfaces [24, 43] or creases [11], which are most commonly represented as triangle meshes or point clouds [22].

The remainder of this paper is organized as follows: After reviewing related work (Section 2), we introduce interactive tools for prefiltering the given data and building a graph on it (Section 3). We then explain spectral graph cuts and their open-box use (Section 4), and present tools for labeling the final clusters and transferring results to similar data (Section 5). After touching upon implementation issues (Section 6), we present three case studies in which our system has been applied successfully to different image analysis tasks (Section 7). After discussing the lessons we learned from building the system and the case studies (Section 8), we conclude (Section 9).

2 Related Work

Our goal differs from semi-automated segmentation [4] or systems for editing segmentation results [41] as they are implemented in Amira (www.amira.com) or MeVisLab (www.mevislab.de). Those methods allow the user to steer or modify each individual result, while we want to help establish segmentation protocols that can consequently be run on similar problems in an automated manner. Paramorama [35] provides a structured framework to navigate and score segmentation results. However, just like Tuner [48], it takes a black box approach.

Previous open-box approaches have provided insight into how Bayesian classifiers [38], neural networks [49], and support vector machines [19, 7] solve specific problems. However, relatively few works exist that try to make the reasoning within machine learning systems understandable to the user, even though this has been highlighted as an important challenge for future visualization systems [25].

To our knowledge, our system is the first open-box approach to spectral clustering [51], which represents the data as a graph and finds a graph cut by using the spectrum (eigenvalues and -vectors) of the graph Laplacian. When applied to classical image segmentation, this technique is often referred to as "spectral segmentation" [27]; therefore, we use both terms interchangeably in this paper. However, spectral clustering should not be confused with image segmentation based on combinatorial min-cuts [6]. Even though that method also uses graph cuts, its objective function and optimization algorithm differ too fundamentally to be unified into the same open-box system.

In this sense, combinatorial min-cuts are one example of a wider range of competing image segmentation algorithms [34] for which open-box variants could be created. For our current work, we selected spectral clustering because it is widely used in medical image analysis, because it allows us to deal with voxel grids, triangle meshes and point sets in a unified manner, and because we believe that showing how its high-dimensional feature space relates to the three-dimensional data domain poses an interesting visualization problem. Even though previous work has mapped three-dimensional streamlines into abstract feature spaces, those were only two- or three-dimensional [8, 21, 40], making them much easier to visualize.

Some of our application examples apply spectral clustering to mesh segmentation. Previous work on this topic [23] has not discussed visual interaction techniques to understand and control the segmentation, nor did it study how parameter settings generalize to similar meshes, or provide any mechanism for transferable labeling.

Our system offers a variant of visual programming, which is, in different forms, implemented in most modern visualization systems, including ParaView (www.paraview.org), SCIRun (www.scirun.org), and VisTrails (www.vistrails.org). Some of them also offer mechanisms for keeping track of user actions, as our system does. However, none of them implement open-box clustering in the sense of our work.

While several other systems support interactive data clustering, including Weka [18], gCLUTO [37], and a recent method for exploring clusters in a large set of subspaces [47], none of them have specific support for spatial 3D data or for spectral clustering.

3 Data Representation and Pre-Filtering

3.1 Graphs as a Unified Data Representation

We make our system general with respect to the representation of the structures that are to be segmented (as point sets, meshes, or grids) by using a graph $G = (V, E)$ as a unified representation. Its vertices V are points in 3D space, i.e., the mesh vertices, particle positions, or voxel positions. These points can have data attributes, such as scalars (e.g., salience of a ridge), vectors (e.g., surface normal), or tensors (e.g., surface curvature).

Each feature type comes with an initial set of edges. This is obvious for the edges of a surface mesh; when data is given on a voxel grid, we introduce edges for all face-connected neighbors. Finally, the particles from [22] have a potential well to control the inter-particle distance, and it is natural to connect points that are located in each others' potential wells with an edge.

The output of our method is a partitioning of the graph vertices into disjoint classes, and a textual label of each class that describes its anatomical meaning (e.g., "frontal lobe"). Our framework also allows the user to discard features without anatomical relevance, which can be interpreted as assigning them to an "outlier" class.

3.2 Building and Manipulating the Graph

The segmentation is based on a similarity $S(\mathbf{p}_i, \mathbf{p}_j) \in [0,1]$ between any two adjacent points $\mathbf{p}_i$ and $\mathbf{p}_j$, which is encoded as an edge weight in the graph, and derived from a distance $D(\mathbf{p}_i, \mathbf{p}_j)$ through

$$S(\mathbf{p}_1, \mathbf{p}_2) = \mathrm{e}^{-\ln(2)\frac{D^2(\mathbf{p}_1,\mathbf{p}_2)}{\sigma^2}}, \tag{1}$$

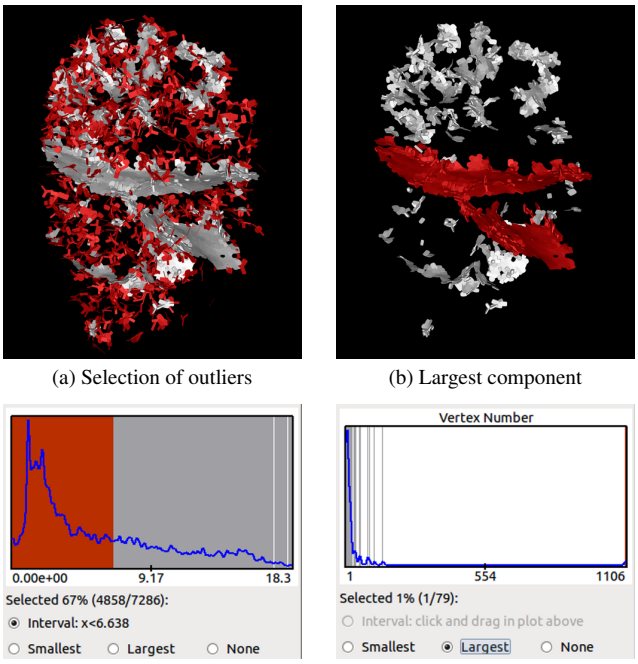
where we include a factor $\ln(2)$ such that the parameter σ marks the boundary between "similar" ($D < \sigma \Rightarrow S > 0.5$) and "dissimilar" ($D > \sigma \Rightarrow S < 0.5$).

The distance D is selected by the user in four steps: First, the attribute from which it should be derived is chosen (e.g., the surface curvature tensor). Second, a choice of distance measures that can be applied to this data type is offered (e.g., the Frobenius norm). Third, the parameter σ is chosen; to suggest a useful default value, our system automatically computes and displays the root mean square of the chosen distance over all edges of the graph. Finally, different similarities S_k, which can be based on different attributes and data types, can be combined into an overall edge weight w_{ij} through multiplication,

$$w_{ij} = \Pi_k S_k(\mathbf{p}_i, \mathbf{p}_j). \tag{2}$$

This amounts to adding the corresponding scaled squared distances $D_k^2(\mathbf{p}_i, \mathbf{p}_j)/\sigma_k^2$ and is quite common in clustering, most frequently to combine data-based with spatial distances [44, 52].

To guide the interactive definition of edge weights, we use kernel density estimation [39] to plot their distribution. An ideal cluster structure is reflected by a clear peak around $S = 1$ (edges within clusters)



(a) Selection of outliers (b) Largest component

(c) Widget used for (a) (d) Widget used for (b)

Fig. 2. The selection widget is a convenient tool for extracting meaningful structure from noisy data (here: the lung fissures from chest CT) in a generalizable way.

and another, often smaller peak around $S = 0$ (edges between clusters). If desired, brushing this plot allows the user to remove edges with a low weight from the graph.

If a disconnected 3D point set is given, or if the initial graph is inappropriate for segmentation, we provide two options for constructing alternative graphs: An ε neighborhood graph connects two points if their distance is smaller than some user-defined radius ε. A k nearest neighbor graph connects two points if at least one of them is among the k most similar vertices of the other, where k is again given by the user. In our current implementation, the ε neighborhood graph is computed based on spatial distances, whereas the k nearest neighbor graph takes into account the custom weights w_{ij} which may combine spatial distances with ones based on more abstract attributes. The user can always revert to working with the original graph.

3.3 The Selection Widget

We have designed a specific widget that can be used to select points or connected components that should be filtered, and components that need further segmentation or should receive their final label.

Points are selected according to some associated quantity that can be chosen by the user. It can be a scalar data attribute of the point $\mathbf{p}_i$, a measure derived from a vector or tensor attribute (such as the trace of the curvature tensor), or its vertex degree d_i, which is defined as the sum of all incident edge weights,

$$d_i = \sum_{j=1}^{|V|} w_{ij}. \tag{3}$$

The selection widget uses a density plot to show the distribution of the chosen quantity. For example, in Figure 2 (a/c), low vertex degree is used to brush and remove noise-related outliers in a particle system that samples ridge surfaces in a chest CT scan, with the goal of segmenting the fissures that separate the lung lobes. Since the particles are a point-based representation, we visualize them using superquadric glyphs [42] of their local covariance tensors to evoke a visual impression of the desired surface, colored red in Figure 2 (b).

The selection in Figure 2 (c) has been made by clicking and dragging on the plot; starting or ending outside the plot defines a half-open interval. Alternatively, the single largest or smallest item can be selected, which is particularly useful when selecting connected components for filtering, clustering, or labeling, as in Figure 2 (d). Scalar attributes of connected components available for selection include vertex and edge numbers, as well as coordinates of the components' centroids and bounding boxes, and invariants of their covariance matrices.

The background color is used to indicate selections (in red), and to ensure that individual data points in sparse regions are not overlooked: If there is at least one data point at any given position, the background is shown in gray rather than white (as can be seen, e.g., in Figure 2 (d)).

The density plot can help the user predict how well a particular selection can be expected to generalize to similar data. For example, the fact that the peak at the left border of the selection widget in Figure 1 (a) is far away from all others suggests that the smallest y coordinate of the cluster bounding box is a reliable criterion to identify the occipital lobe of the brain. In Section 5.2, we will use such reasoning to automatically propose good selection criteria for given clusters.

4 Open-Box Spectral Graph Cuts

Given a connected component (V,E) of the weighted graph from the previous section, spectral clustering partitions the vertices V into disjoint classes such that edges within a class have high weights, whereas edges between classes have low weights. This captures meaningful structure in the data by grouping together similar points, where an application-specific notion of similarity is encoded in the user's choice of edge weights.

4.1 A Brief Introduction to Spectral Clustering

We give a brief introduction to spectral clustering to make this paper self-contained. The interested reader is referred to [51, 13] for a more detailed discussion.

If $|V|$ is the number of vertices, the graph can be represented as a symmetric $|V| \times |V|$ adjacency matrix $\mathbf{W}$, whose entries w_{ij} equal the edge weights defined in Equation (2). Moreover, the degree matrix $\mathbf{D}$ is the diagonal matrix with elements d_i from Equation (3). From these matrices and the identity $\mathbf{I}$, the unnormalized graph Laplacian $\tilde{\mathbf{L}}$ and the normalized graph Laplacian $\mathbf{L}$ are defined as

$$\tilde{\mathbf{L}} = \mathbf{D} - \mathbf{W} \quad \text{and} \quad \mathbf{L} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{W}. \tag{4}$$

The set of edges between two vertex classes V_1 and V_2 is called a "cut", and the sum of their weights is denoted by $\mathrm{cut}(V_1,V_2)$. The basic idea of spectral clustering is to use the spectrum (i.e., the eigenvectors and eigenvalues) of $\tilde{\mathbf{L}}$ or $\mathbf{L}$ to find a cut that separates the graph G into k subgraphs, thus arriving at k clusters.

The unnormalized Laplacian has been used in the context of graph partitioning for almost 40 years [10, 12]. For the bisection of V into V_1 and V_2, Hagen and Kahng [17] have shown that the unnormalized Laplacian is related to the

$$\mathrm{RatioCut} = \frac{\mathrm{cut}(V_1,V_2)}{|V_1|} + \frac{\mathrm{cut}(V_1,V_2)}{|V_2|}. \tag{5}$$

In contrast to the minimum cut, which bisects a graph by removing a set of edges with minimum weight and frequently cuts off a single vertex or a very small part of the graph, RatioCut encourages more balanced cuts. Unfortunately, it is NP hard to find the partition (V_1,V_2) with the optimal RatioCut. The eigenvector corresponding to the second smallest eigenvalue of $\tilde{\mathbf{L}}$ can be used to compute an approximate solution, in which real numbers replace binary cluster labels, and subsequent thresholding is used to obtain a hard assignment to V_1 or V_2.

Alternatively, Shi and Malik [44] propose the normalized cut criterion,

$$\mathrm{NCut} = \frac{\mathrm{cut}(V_1,V_2)}{\mathrm{vol}(V_1)} + \frac{\mathrm{cut}(V_1,V_2)}{\mathrm{vol}(V_2)}, \tag{6}$$

where the volume $\mathrm{vol}(V)$ is defined as the sum of the vertex degrees d_i within V. An approximate version of NCut is computed in a similar manner as RatioCut, with the normalized Laplacian $\mathbf{L}$ taking the place of its unnormalized counterpart.

Both RatioCut and NCut encourage dissimilar data to be placed in different clusters, and if all vertices have the same degree d_i, both criteria lead to the same optimizer (V_1, V_2). When the two differ, RatioCut tries to balance the cut in terms of vertex numbers, while NCut puts more emphasis on within-cluster similarity. Even though this makes NCut superior for many practical tasks, the decision is ultimately task-dependent, and our framework offers both options.

Weighted kernel k-means provides a different algorithmic approach to optimizing the objective functions presented above [3, 13]. This alternative is faster, since it does not require eigenvector computation, but it is sensitive to its initialization and it is more prone to local optima, which has been addressed by a multilevel implementation [9]. In our work, we make use of the more traditional spectral approach, since the feature space in kernel-based methods is implicit and often infinite-dimensional, which makes it less suitable for visualization within an open-box framework.

4.2 Treating Multiple Clusters

In its simplest form, spectral clustering uses the second smallest eigenvector of the graph Laplacian to partition a graph into two pieces. There exist two common strategies to find more than two clusters. The hierarchical approach applies the same kind of bisection to the clusters obtained in the previous step, until some stopping criterion is reached. It is a drawback of this approach that it does not naturally handle cases in which the Laplacian has repeated eigenvalues, so that the second smallest eigenvector is part of a higher-dimensional eigenspace.

An alternative approach is to use a naïve clustering technique such as k-means to look for clusters in the so-called spectral embedding: Each of the $|V|$ coefficients in the eigenvectors of $\mathbf{L}$ corresponds to one of the graph vertices, and the spectral embedding of a vertex is the d-dimensional point whose coordinates are the corresponding coefficients of the smallest d eigenvectors. This abstract feature space enhances the cluster structure in the data, which makes it possible to identify the clusters with a simple clustering technique such as k-means, even if that technique is insufficient when applied to the original data [27, 29].

As an alternative to the spectral embedding, we also offer a variant of the commute time embedding. The commute time between two vertices v_1 and v_2 is defined as the expected time a random walker needs to travel from v_1 to v_2 and back, where transition probabilities account for the edge weights. The commute time is lowest if there are several efficient paths between v_1 and v_2, so it encourages clustering together nodes that are strongly connected in more than one way.

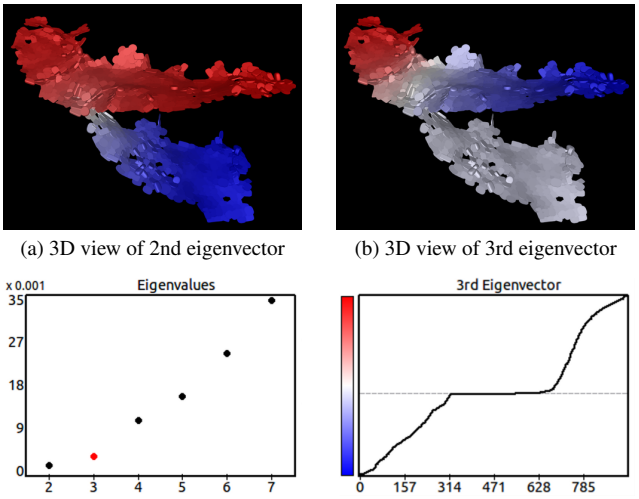
The commute time embedding uses the Laplacian spectrum of a graph $G = (V,E)$ to assign to each vertex a position in $\mathbb{R}^{|V|}$ such that Euclidean distances in $\mathbb{R}^{|V|}$ equal the commute time [36]. Even though computing the exact commute time requires all eigenvectors, their significance decreases with increasing associated eigenvalues. Since computing all eigenvectors is impractical when $|V|$ is large, we only compute the d most significant coordinates, associated with the d smallest eigenvalues of the Laplacian.

Even though all three embeddings can be justified theoretically, it is difficult to predict which of them will work best in a given application, which feature space dimension d is ideal, and whether or not a hierarchical clustering strategy outperforms simple k-means on the spectral embedding. Our system encourages interactive exploration of these alternatives by visualizing the resulting feature spaces.

4.3 Visualizing the Spectrum

If there are k well-defined clusters in the data, spectral graph theory predicts a gap between the kth smallest and the $k+1$st smallest eigenvalue of $\mathbf{L}$. Plotting the leading $n > k$ eigenvalues allows identification of this spectral gap. In Figure 3 (c), the fact that the gap between the third and the fourth eigenvalue is much larger than between the second and the third one suggests that for the lung fissures, three is a reasonable number of clusters. The first eigenvalue is omitted from the plot, since it is always 0, by definition of $\mathbf{L}$.

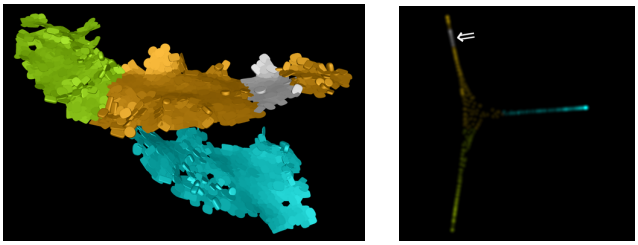
Clicking on an eigenvalue plots the coefficients of the corresponding eigenvector, sorted by their signed values, as shown in Figure 3 (d).



(a) 3D view of 2nd eigenvector (b) 3D view of 3rd eigenvector

(c) Plot of smallest eigenvalues (d) Plot of 3rd eigenvector

Fig. 3. Interactively linked 2D plots and 3D color codings of eigenvectors help decide the number of clusters and convey how the eigenvectors encode the cluster structure.



(a) (b)

Fig. 4. Brushing (gray regions in both subfigures) connects the 3D view (a) to a plot of the graph embedding (b).

In spectral clustering, each eigenvector coefficient corresponds to one of the clustered points. In order to better convey this correspondence, we color code the coefficients in the 3D view, as shown in Figure 3 (b). This reveals that in our example, the two increasing and the central, almost constant part of the line plot correspond to the three segments of the fissure. Figure 3 (a) shows the coloring that would result from clicking on the second eigenvalue, whose eigenvector separates the horizontal part of the fissure from the diagonally descending part.

Many spectral clustering approaches threshold eigenvector coefficients to partition the points into two classes. The dashed line in the eigenvector plot indicates zero, which is a commonly used threshold [44]. As an additional feature which is not shown in the Figure, we allow the user to explore the effect of different thresholds by clicking on the eigenvector plot. In addition to the color coding, this interaction can help decide which eigenvectors to use for the final clustering.

4.4 Visualizing the Clustering

Once the dimensionality of the embedding has been decided, k-means is applied to the embedded points to obtain the final clustering. In order to work around potential suboptima [31], this step is repeated several times, each time selecting the seeds uniformly at random, and retaining the result that produced the minimum mean squared distance between points and their cluster center.

In the 3D view, the clustering is shown by pseudocolors, Figure 4 (a). As a visual way of checking that the embedding has led to a clear cluster structure and that k-means has captured it, the side panel shows a scatterplot of the points in the spectral embedding (b). In this simple example, the feature space is two dimensional, so it is shown fully in the plot. In more realistic scenarios, the spectral embedding has more than two dimensions. In that case, we offer a "grand tour"

[2] through them, i.e., we create a continuous series of 2D projections in which the user can navigate by going forward or backward at an adjustable speed. Initially, the axes are aligned with the second and third eigenvectors, which provide the most salient clustering information.

The number of data points in most realistic scenarios is so large that it would clutter a traditional scatterplot. Therefore, we instead splat small Gaussian kernels for each point, and accumulate their contributions, using OpenGL's point sprites extension. The correspondence between points in the abstract feature space and in the 3D view is clarified by using the same color for corresponding points. As an additional interaction technique, we allow the user to directly probe the correspondence by brushing the scatterplot. Such brushing has created the gray region in Figure 4, marked by an arrow in (b). It links the two views in an intuitive manner.

5 Transferring User Actions to Similar Data

5.1 Rule Files

Since it is our goal to help the analyst develop segmentation strategies that can be successfully transferred to similar data, our system keeps track of all user actions in a so-called rule file. Rule files record all user interactions that are relevant to the final result. In order to make them easy to edit and to process with other software, they are stored in XML format. Examples can be seen in the supplementary material, where the rule files for all experiments from Section 7 are provided to guarantee reproducibility of our results.

Rule files consist of three sections: "InputData" describes the types and names of data attributes that were present; rules can only be transferred to data sets that provide the same attributes. "Labels" lists all textual labels that were used to describe semantic structures in the data, and the color codes that were assigned to them. Finally, "Rules" is the complete list of actions that have been taken to achieve the final labeling, including all parameters. This includes filtering by attributes or vertex degree, changing the graph type, weighting and removing edges, clustering, and filtering or labeling connected components.

Unlike VisTrails, which aims to keep a complete account of how a workflow evolved [45], our priority is to keep the rule files concise and fast to apply. Therefore, recording of exploratory user actions is deferred until it becomes clear whether they will really affect the final result. In particular, changes made to the edge weights are only recorded when an action such as filtering or clustering depends on them, and additional clustering parameters, such as type and dimensionality of the embedding, are only recorded once the user is satisfied with the result and has clicked a button to accept it.

5.2 Proposing Selection Rules

The most intuitive way to select components for labeling, filtering, or hierarchical subdivision is by clicking on them in the 3D view. Unfortunately, it is not clear how this type of user interaction should be transferred to other datasets. Therefore, our system provides a mechanism for proposing formal selection rules based on an intuitive 3D selection. Even though implemented differently, this is somewhat similar in spirit to a system by Fuchs et al. [15], which searches for simple, human understandable rules that result in a spatial selection similar to the one made by the user.

Criteria that can be used for automated selection consist of an attribute (such as cluster size, shape, or spatial position) and a property of that attribute (such as a specific coordinate of the position). Our mechanism for proposing rules works by assigning a score to each available criterion. Alternatively, criteria can also be browsed and selected manually using drop-down boxes and the density plot that are visible in the right panel of Figure 1 (a).

Scoring is based on the insight that if a given component i is very dissimilar from all other components with respect to some criterion c (e.g., much larger than all others), that criterion is likely to be well-suited for reliable selection. Therefore, we define the score as

$$s_i(c) := (c_{\max} - c_{\min}) \left(\frac{1}{\bar{d}_i(c)} + \frac{1}{\underline{d}_i(c)} \right), \tag{7}$$

where lower values of s indicate more reliable criteria. $c_{\max}$ and $c_{\min}$ denote the largest and smallest value of c over all components, and are used to normalize all scores to a common scale. $\bar{d}_i(c)$ and $\underline{d}_i(c)$ are the distances of component i to the nearest larger or smaller component with respect to criterion c, respectively. If no such nearest component exists because i attains the minimum or maximum value of c, the inverse of the respective distance $\underline{d}_i(c)$ or $\bar{d}_i(c)$ is taken to be zero; if another component exists with the same value of c, that criterion is excluded (formally, s is set to infinity).

All computed scores are sorted and the five best criteria according to this ranking are presented for selection by the user. Involving a human is useful because in many cases, several criteria rank similarly, and the human expert often has additional insight on which of them can be expected to generalize best to new data (e.g., if spatial coordinates can be relied on, or if orientation with respect to the given coordinate system is arbitrary in the given application).

Often, the desired component is the smallest or largest one with respect to the selected criterion. In this case, our system will simply look for the smallest or largest component with respect to the same criterion in order to make an analogous selection in new data. Otherwise, the system will look for the component whose value is closest to the one from the example dataset based on which the rule file was generated.

6 Implementation and Performance

6.1 General Design of our System

Figure 1 (a) shows the basic layout of our user interface. A toolbar at the bottom leads the user through the subsequent steps of loading and filtering the data, modifying its representation as a graph, clustering, and labeling the resulting regions. Finally, it allows inspecting the rule file, saving it to disk, and applying it to different data sets. Each tab in the toolbar reveals a different set of two-dimensional data views in the right pane. Linking and brushing connect them to the 3D data, which is visible at all times.

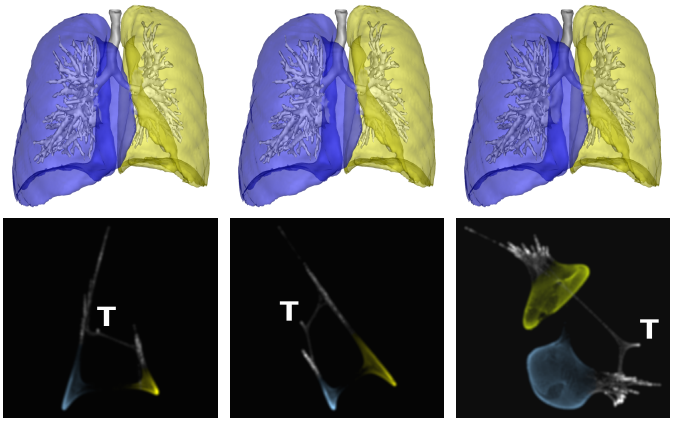
The system has been built using C++ with QT (qt.nokia.com) for the graphical user interface, OpenGL for 3D graphics, and Teem (teem.sf.net) for isosurface extraction, its implementation of particle systems [22], and generation of tensor glyphs. Efficiently computing the n smallest eigenvectors of large sparse matrices is made possible by so-called Krylov subspace methods [46], whose implementation in SLEPc (www.grycap.upv.es/slepc) [20] we use.

While we could use the XML support built into QT, several widgets, including the selection widget from Section 3.3, the plotting widgets in Figure 3, and the brushable grand tour widget from Section 4.4, have been implemented from scratch and specifically for this project.

6.2 Computational Performance

The presented experiments were done on a standard workstation (2.7 GHz, 8 GB of RAM), and most operations happened at interactive frame rates. Exceptions were construction of the 20 nearest neighbor graph for the putamen (Section 7.3, around $8,000$ vertices, $105,000$ edges), which requires computation of all pairwise distances and took around 4 sec. For this example, eigenvector computation was still interactive (0.2 sec). However, computing 10 eigenvectors for the brain mesh (Section 7.2, around $23,000$ vertices, $68,000$ edges) took 6 sec, and 5 eigenvectors for the lung meshes (Section 7.1, around $60,000$ vertices, $175,000$ edges) took 50 sec. In order to handle even larger problem sizes within an acceptable time, we plan to employ strategies such as approximating eigenvectors based on a randomly chosen subsampling [14] or pre-grouping [53] in future versions of our system.

Since eigenvector computation for most non-trivial clustering tasks is non-interactive, we provide types of visual feedback for many settings that do not require this computation; an example are density plots of the defined distance measure, which can be brushed to identify regions of low edge weights in the 3D view, where the clustering is more likely to place boundaries. On the other hand, once the embedding has been computed, we provide an interface for changing parameters such as the cluster number without requiring its recomputation each time, which would be unavoidable if we treated the whole clustering algorithm as a monolithic black box.



(a) RatioCut (b) NCut (c) Commute Time

Fig. 5. Since different embeddings can lead to similar segmentation results (top row), inspection of the spectral embedding (bottom row) is a valuable guide for choosing between them.

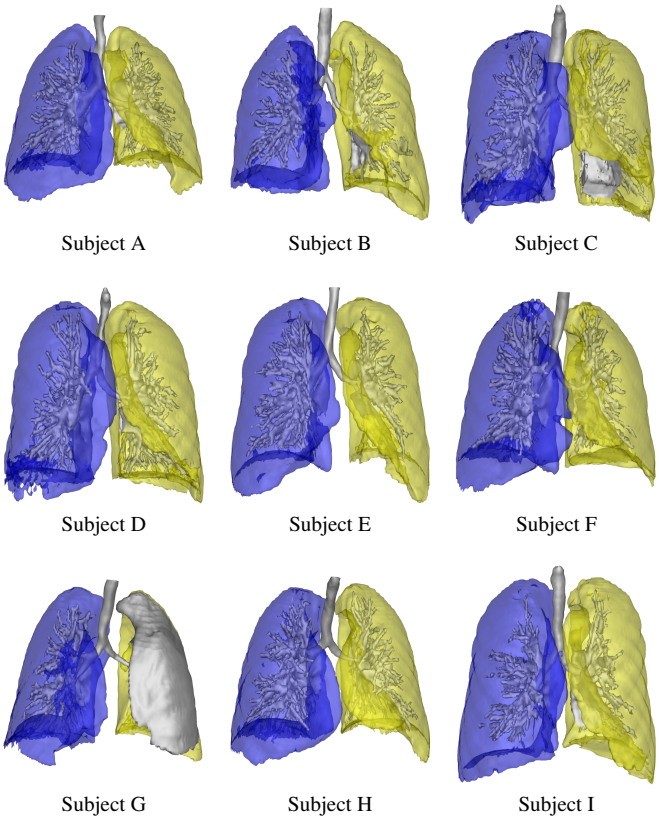


Fig. 6. Automatically transferring the result from Figure 5 to nine other subjects produces many useful results, and highlights cases (such as Subject G) for which segmentation is most challenging.

7 Case Studies

7.1 Recognizing Airways in Chest CT

In chest CT, the bronchial tree and the outer surface of the lungs are given by a single isosurface. As an initial task, we tried to differentiate and label those surface parts, so that the outer lung surface can be rendered semi-transparently and the bronchial tree is revealed.

Starting with the first subject, we quickly obtained a useful spectral segmentation by using edge weights that depend on the ℓ_2 differences in surface normals. As shown in the upper row of Figure 5, trying different embeddings led to subtle differences in the segmentation, mostly around the point at which the bronchial tree enters the lungs. The fact that these differences are small makes it difficult to decide on a particular type of graph based on the final segmentation result alone. Therefore, we instead selected the embedding by exploring the corresponding up to five-dimensional feature spaces with our brushable grand tour widget.

The results are shown in the bottom row of Figure 5. In all cases, the overall structure of the surface, which consists of trachea (marked by a T), bronchial tree (white) and left/right lung surface (yellow/blue), is clearly reflected in the spectral embedding. The embeddings corresponding to the approximated RatioCut and NCut are quite similar; the commute time embedding more clearly separates individual branches of the bronchial tree. Watching the grand tour animations suggested that the desired structures are separated most clearly in the commute time embedding. After segmentation, clicking on the lung surfaces and using the selection rules proposed by our system made it straightforward to label the right (blue) and left (yellow) lung. A single label ("airway"; gray) was assigned to all remaining classes.

We applied the resulting rule file to the scans of nine other patients. Figure 6 faithfully reflects the potential and limitations of our system by presenting the results on all of them. A useful segmentation is achieved in almost all cases. In general, we should not expect that it will be possible to develop a segmentation strategy based on a single example that will be fully reliable even under large anatomical variations and pathology, like they are found in this patient database. Rather, it will be unavoidable to base the development of more general and robust methods on more than a single example. In this context, the rule files generated by our system can help to build a ground truth database by including the examples (such as subjects A, D, E, F, and H) in which the transfer was fully satisfactory, and to identify cases (such as subject G) in which segmentation is most challenging.

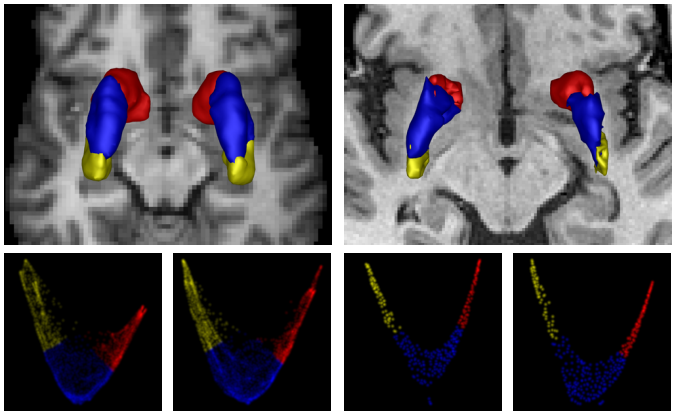
7.2 Recognizing Lobes of the Brain

For a second case study, we extracted the white matter outline as an isosurface from the output of a tissue segmentation algorithm that combines information from coregistered structural T1, T2 and PD MR images [26]. Our goal was to segment this white matter outline, and to assign its regions to the five different lobes of the brain, as shown in Figure 1.

Even though the lobes are named after the bones that surround them, their overall structure can be recovered from the sulci of the cortex, which is reflected in the white matter. Therefore, we defined edge weights that penalize large positive mean curvature; since the isosurface normals point into the brain, this corresponds to the valleys of the surface when viewed from outside.

In this task, we found it impossible to achieve a satisfactory segmentation in a single step, despite experimenting with different clustering parameters. Therefore, we let the system guide us towards a hierarchical approach: Using the tools from Section 4.3 to inspect the Laplacian spectrum initially suggested the presence of five clusters; however, they did not yet correspond to the five desired lobes, but rather contained an oversegmentation of the frontal lobe (red in Figure 1) into two parts, which were easily joined back together by assigning the same label. At this time, the brainstem (orange) and temporal lobe (blue) remained a single region, which was easily separated into the desired parts in a hierarchical second clustering step. Again, the commute time embedding turned out to be the most reliable choice for segmentation. The mechanism from Section 5.2 made it very easy to find rules that select regions for labeling or further subdivision.

We confirmed that the rules we found in this semiautomated manner can be successfully transferred to ten other subjects, which are shown in Figure 1 (b). Again, we show the data from all subjects that were included in our experiment, to provide a realistic impression of how well our system performs. Consulting a neurologist confirmed that the overall assignment is correct, even though the exact boundaries between the regions are displaced in some cases, especially the temporal lobe in Subject H.



Subject from Site A Subject from Site B

Fig. 7. Good agreement is found between the putamen segmentations achieved for two different subjects, measured and preprocessed at two independent sites. However, the spectral embedding suggests that the clusters are not sharply separated.

7.3 Recognizing Subgroups of the Putamen

In our first two case studies, we transferred segmentation parameters to similar data from the same site. Often, reproducing results across sites is even more challenging, especially in diffusion tensor imaging (DTI) [32]. In a third case study, we attempt such a transfer across sites. The specific task is to use information from DTI to segment gray matter structures, which appear homogeneously in standard MRI. The feasibility of such segmentations was first presented on the thalamus [52]; our experiment will focus on the putamen, a gray matter structure that is part of the basal ganglia [1].

The DTI data is given on a voxel grid, and we only consider voxels within a putamen mask provided by our collaborators. In our analysis, we first used the selection widget from Section 3.3 to filter out voxels with large mean diffusivity or large fractional anisotropy [5], which indicate significant partial voluming with structures outside the putamen (i.e., cerebrospinal fluid or white matter).

In our experiments, we found that adding edges only for face-connected voxel neighbors resulted in an undesirably strong spatial regularization of the final segmentation. Therefore, we used the graph manipulation described in Section 3.2 to define a 20 nearest neighbor graph, based on the Frobenius norm of the tensor deviatoric as the distance measure. Adding a very slight spatial component to the distance ($\sigma = 15\,\text{mm}$) did not change the overall structure of the segmentation, but reduced the number of isolated voxels that would otherwise clutter the visualization. In accordance with the results of Anwander et al. [1], the spectrum of the normalized graph Laplacian $\mathbf{L}$ suggested the presence of three clusters, which were extracted in a single step based on the two-dimensional spectral embedding induced by $\mathbf{L}$. This corresponds to the normalized cut criterion in Eq. (6); the two other embeddings produced very similar results.

Figure 7 shows the result, and confirms that the same rule file could be applied to analyze the putamen in the opposite hemisphere, as well as in a diffusion tensor dataset from another subject, acquired at a different site with a different measurement setup, a different spatial resolution, a different preprocessing pipeline, and a different way of defining the putamen mask [33]. To convey a better spatial impression, clusters are visualized as smoothed [50] isosurfaces after rasterizing the labels into a 3D grid.

In both cases, a qualitatively similar segmentation is obtained. This is already a remarkable result, since the only previous work that has presented a DTI-based segmentation of the putamen performed a group average segmentation of 14 subjects [1], and did not attempt segmentation in individuals. Our result supports the finding that DTI data can be used to segment the putamen into three parts. However, the spectral embedding, shown in the bottom row of Figure 7, suggests that the exact boundaries between the regions are somewhat arbitrary, since the data does not exhibit clearly separated clusters.

Table 1. The dice indices in this table show that when segmenting the putamen, the results of our method are much more stable under dilation of the mask than the ones from the previous state of the art.

Region	k means [1]	Our method
Anterior	0.833	0.911
Central	0.502	0.919
Posterior	0.681	0.947

We also obtained quantitative results by evaluating the robustness of our labelings with respect to small inaccuracies in the putamen mask. For this, we applied morphological dilation to enlarge the mask by one voxel along its boundary, and applied the same rule file to the modified input. An algorithm that handles this perturbation well will create a new labeling that is very similar to the original one. Table 1 shows that in this regard, our method performs much better than the one that was previously used for the same task [1].

The dice index which is reported for all three regions is defined as $\text{dice} = 2|X \cap Y|/(|X| + |Y|)$, where X is the set of voxels assigned to the respective region originally, and Y is the set of voxels assigned to it after the perturbation. For this comparison, the result on the enlarged mask has been restricted to the original (smaller) mask. The correct labeling of the regions as "anterior" (red), "central" (blue) and "posterior" (yellow) has been transferred automatically from the training example by our method, whereas it had to be assigned manually when using the previous method.

8 Discussion

It is widely accepted that no single clustering or image segmentation algorithm works well in all scenarios [16]. Even after decades of research, adapting such methods to specific application problems remains a process of trying out different strategies and parameter values. Our system does not attempt to change this fact. Rather, it implements a multitude of strategies that support and guide the user in this trial-and-error process. Examples of such guidance are plots of distributions and display of summary statistics when defining distance measures and graphs, computation of sensible default values whenever possible, brushing of affected data points when defining rules for filtering, tools for exploring the spectral embedding, and proposal of reproducible selection rules based on clicks in the 3D view.

At the current stage, it is not our main goal to design a graphical user interface that would allow domain experts without a strong technical or theoretical background to use spectral clustering. Rather, our current interface caters to experts who are used to performing very similar adjustments by repeatedly running a segmentation algorithm and observing the changes in the results. We have previously worked with spectral clustering and other segmentation methods in this more traditional way ourselves, and found that our interactive system greatly reduces the time required to achieve an initial usable result, and encourages further exploration. Even though the guidance provided by our system is valuable, it pre-supposes a basic understanding of how the individual steps in spectral clustering work and what the relevant parameters mean.

8.1 Lessons Learned from the Case Studies

The main challenge in turning spectral clustering into an open-box tool was to visualize the potentially high-dimensional spectral feature space and to link it back to the original three-dimensional data.

Showing density-based scatter plots of the spectral feature space and, if needed, creating a continuous series of projections using the grand tour [2], proved very helpful to inspect the two- to five-dimensional feature spaces from the chest CT and the putamen DTI case studies. There was a clear correspondence between salient features in the spectral space and meaningful spatial structures, and the exact relationship was easily established using the brushing offered by our system. Viewing the spectral embedding clarifies how well the clusters are separated in feature space. This is helpful to decide on

the type of embedding that is best suited for a given problem, since it predicts how likely we are to achieve a similar segmentation on data from a different subject. Such a prediction is much more difficult to make based on the segmentation result alone.

In the fifteen-dimensional feature space that we used for brain lobe segmentation, the plots provided by the grand tour became less informative. In this case, viewing plots of the individual eigenvectors, and color mapping their coefficients in the 3D view, provided a more useful way to explore the embedding, and successfully guided us towards a hierarchical clustering strategy that was not obvious from looking at the three-dimensional data alone.

In all case studies, we made extensive use of the mechanism for proposing selection rules from interactive 3D selections, as described in Section 5.2. In all experiments, one of the top five candidates quickly stuck out as resulting in a good selection rule. Our initial choice always generalized well to new data and did not have to be revised to achieve the presented results.

8.2 Relationship to "Programming With Example"

The rule file which is generated by our system can be regarded as a computer program in a high-level language specialized to performing filtering, clustering, and labeling of three-dimensional data. Even though this file is easily modified in any text editor, it is not originally generated by typing text. Rather, it is an automatic record of the interactions with a graphical user interface.

In this sense, our system follows the paradigm of "programming with example", in which the user demonstrates to the computer how to solve a given problem by working through a specific example in a manual or semiautomated manner. According to Myers' taxonomy [28], "programming *with* example" means that in order to solve a new problem, the computer will exactly follow the steps taken by the user in the demonstration. Since developing segmentation methods that work reliably under strong anatomical variation and pathology requires working with more than a single example, a useful future extension of our system would be towards "programming *by* example", where machine learning techniques are used to automatically infer a more general program from observing the user strategy on several examples. However, the results from our case study show that the rule files generated by our system are already useful in their current form.

8.3 Informal Expert Feedback

We gave live demonstrations of our system and received informal feedback from a PhD student in computer science with profound knowledge in graph embeddings, and from a neurologist at the university hospital. The student emphasized that recent theoretical progress on clustering did not remove the need to experiment with different methods and parameters in practical applications, as it is facilitated by our framework. He had not been familiar with the grand tour [2] and found our framework very useful to explore high-dimensional graph embeddings. He suggested several additional features, some of which are part of our current version.

The neurologist was impressed by the ability to transfer the manually assigned labels from an example dataset to other data, and expressed the hope that such a system might alleviate the burden of repetitive manual image processing that is still part of image-based clinical studies. He found it interesting to interactively try different methods for DTI-based clustering of gray matter structures, and the visualizations of the spectral embedding were helpful in discussing their respective potential and limitations.

9 Conclusion

Similar to natural language analysis [30], image analysis is a problem that, in the foreseeable future, will continue to benefit from closely integrating automated methods with feedback from human experts. Despite this, we found very little work on this problem in the Visual Analytics literature.

The most closely related previous works [48, 35] have proposed black-box approaches; the results achieved with our novel system suggest that an open-box approach, which makes the impact of user choices on internal key elements of the algorithm transparent, is a worthwhile alternative with the following advantages:

1. In many cases, open-box approaches can provide useful visual feedback without having to compute a final segmentation. For example, plotting histograms of distance measures provides an immediate impression of what range of parameter values is sensible, which reduces the number of times the potentially quite time consuming segmentation method needs to be run. Similarly, exploring the eigenvectors of the graph Laplacian allows us to make an informed choice on how many of them should be used, even before computing the final segmentation.

2. While previous work [48, 35] has concentrated on tuning a fixed set of continuous parameters, our framework also supports decisions that cannot be modeled in this way, but are still crucial in many segmentation tasks. Those include selecting among potential distance measures and performing hierarchical clustering.

3. We do not require a formal, predefined ground truth, but rather rely on human judgement on whether or not a proposed grouping of the data is meaningful. From the user's perspective, we consider this a key factor in deciding between an open-box or a black-box system: If ground truth is available, a black-box method, or even fully automated cross-validation, can be used to fix parameters. Otherwise, one might prefer the more exploratory approach facilitated by an open-box system. Producing ground truth manually can be cumbersome especially in 3D image analysis; among other benefits, the rule files generated by our system can help with semi-automated production of a ground truth database.

Other features of our system, which go beyond the functionality found in Tuner [48] and Paramorama [35], are mechanisms for filtering and labeling data in a way that is convenient and generalizes to similar data. Like these previous approaches, we currently only visualize the impact of segmentation parameters on the data of a single subject. In our case studies, this already allowed us to produce rules that could be successfully applied to many similar datasets. Finding techniques to interactively explore the effect of segmentation strategies on a whole ensemble of datasets will be a challenging task for future work.

Finally, we would like to point out that even though our system is specialized to three-dimensional image analysis, spectral clustering itself is fully general, and the proposed techniques for turning it into an interactive open-box method could potentially also benefit other clustering tasks in Visual Analytics.

Acknowledgments

We would like to thank Hans J. Johnson (University of Iowa) and Alfred Anwander (MPI CBS, Leipzig, Germany) for providing the brain MR data used in this work; data acquisition at UIowa was funded by the grants NS054893 and NS40068. Lung CT images were taken from the VIA/I-ELCAP public lung database (Cornell Medical Center).

We are grateful to Tobias Schmidt-Wilcke (University Hospital Bergmannsheil, Bochum, Germany), Morteza Alamgir (University of Hamburg, Germany), Holger Theisel (University of Magdeburg, Germany), Charl Botha (vxlabs), and Daniel Keim (University of Konstanz, Germany) for providing valuable feedback.

The first author did parts of this work while he was with the MPI for Intelligent Systems, Tübingen, Germany.

References

[1] A. Anwander, T. Knösche, and S. A. Kotz. Sub-compartments of the human putamen: A DTI study. In *Proc. Human Brain Mapping*, page 1329, 2010.

[2] D. Asimov. The grand tour: A tool for viewing multidimensional data. *SIAM J. Scientific and Statistical Computing*, 6(1):128–143, 1985.

[3] F. Bach and M. Jordan. Learning spectral clustering. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Proc. Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2004.

[4] W. A. Barrett and E. N. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1(4):331–341, 1997.

[5] P. J. Basser and C. Pierpaoli. Microstructural and physiological features of tissues elucidated by quantitative-diffusion-tensor MRI. *Journal of Magnetic Resonance, Series B*, 111:209–219, 1996.

[6] Y. Boykov and G. Funka-Lea. Graph cuts and efficient n-d image segmentation. *Int'l J. Computer Vision*, 70(2):109–131, 2006.

[7] D. Caragea, D. Cook, H. Wickham, and V. Honavar. Visual methods for examining SVM classifiers. In S.J. Simo et al., editor, *Visual Data Mining*, volume 4404 of *LNCS*, pages 136–153. Springer, 2008.

[8] W. Chen, Z. Ding, S. Zhang, A. MacKay-Brandt, S. Correia, H. Qu, J. A. Crow, D. F. Tate, Z. Yan, and Q. Peng. A novel interface for interactive exploration of DTI fibers. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization)*, 15(6):1433–1440, 2009.

[9] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[10] W. E. Donath and A. J. Hoffman. Lower bounds for the partitioning of graphs. *IBM J. of Research and Development*, 17(420–425), 1973.

[11] D. Eberly. *Ridges in Image and Data Analysis*, volume 7 of *Computational Imaging and Vision*. Kluwer Academic Publishers, 1996.

[12] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(2):298–305, 1973.

[13] M. Filippone, F. Camastra, F. Masulli, and S. Rovetta. A survey of kernel and spectral methods for clustering. *Pattern Recognition*, 41(1):176–190, 2008.

[14] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 26(2):214–225, 2004.

[15] R. Fuchs, J. Waser, and M. E. Gröller. Visual human+machine learning. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization)*, 15(6):1327–1334, 2009.

[16] I. Guyon, U. von Luxburg, and R. C. Williamson. Clustering: Science or art? In *NIPS Workshop Clustering: Science or Art*, 2009.

[17] L. Hagen and A. B. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design*, 11(9):1074–1085, 1992.

[18] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[19] L. Hamel. Visualization of support vector machines with unsupervised learning. In *Proc. IEEE Symp. on Computational Intelligence and Bioinformatics and Computational Biology (CIBCB)*, pages 1–8, 2006.

[20] V. Hernández, J. E. Román, A. Tomás, and V. Vidal. Krylov-schur methods in SLEPc. Technical report, Universidad Politecnica de Valencia, 2007.

[21] R. Jianu, Ç. Demiralp, and D. H. Laidlaw. Exploring 3D DTI fiber tracts with linked 2D representations. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization)*, 15(6):1449–1456, 2009.

[22] G. Kindlmann, R. San José Estépar, S. M. Smith, and C.-F. Westin. Sampling and visualizing creases with scale-space particles. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Visualization)*, 15(6):1415–1424, 2009.

[23] R. Liu and H. Zhang. Segmentation of 3D meshes through spectral clustering. In *Pacific Conf. on Computer Graphics and Applications*, pages 298–305, 2004.

[24] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. ACM SIGGRAPH*, pages 163–169, 1987.

[25] K.-L. Ma. Machine learning to boost the next generation of visualization technology. *IEEE Computer Graphics and Applications*, 27(5):6–9, 2007.

[26] V. A. Magnotta, G. Harris, N. C. Andreasen, D. S. O'Leary, W. T. Yuh, and D. Heckel. Structural MR image processing using the BRAINS2 toolbox. *Computerized Medical Imaging and Graphics*, 26:251–264, 2002.

[27] M. Meila and J. Shi. A random walks view of spectral segmentation. In *Proc. Int'l Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2001.

[28] B. A. Myers. Visual programming, programming by example, and program visualization: A taxonomy. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 59–66, 1986.

[29] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 849–856. MIT Press, 2002.

[30] D. Oelke, D. Spretke, A. Stoffel, and D. A. Keim. Visual readability analysis: How to make your writings easier to read. *IEEE Trans. on Visualization and Computer Graphics*, 18(5):662–674, 2012.

[31] J. Peña, J. Lozano, and P. Larrañaga. An empirical comparison of four initialization methods for the *k*-means algorithm. *Pattern Recognition Letters*, 20:1027–1040, 1999.

[32] A. Pfefferbaum, E. Adalsteinsson, and E. V. Sullivan. Replication of diffusion tensor imaging measurements of fractional anisotropy and trace in brain. *Journal of Magnetic Resonance Imaging*, 18:427–433, 2003.

[33] S. Powell, V. A. Magnotta, H. Johnson, V. K. Jammalamadaka, N. C. Andreasen, and R. Pierson. Registration and machine learning based automated segmentation of subcortical and cerebellar brain structures. *NeuroImage*, 39(1):238–247, 2008.

[34] B. Preim and D. Bartz. *Visualization in Medicine. Theory, Algorithms, and Applications*. Morgan Kaufmann, 2007.

[35] A. J. Pretorius, M.-A. Bray, A. E. Carpenter, and R. A. Ruddle. Visualization of parameter space for image analysis. *IEEE Trans. on Visualization and Computer Graphics*, 17(12):2402–2411, 2011.

[36] H. Qiu and E. R. Hancock. Clustering and embedding using commute times. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 29(11):1873–1890, 2007.

[37] M. Rasmussen and G. Karypis. gCLUTO – an interactive clustering, visualization, and analysis system. Technical Report TR 04–021, University of Minnesota, 2004.

[38] P. Rheingans and M. desJardins. Visualizing high-dimensional predictive model quality. In *Proc. IEEE Visualization*, pages 493–496, 2000.

[39] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *Annals of Mathematical Statistics*, 27(3):832–837, 1956.

[40] C. Rössl and H. Theisel. Streamline embedding for 3D vector field exploration. *IEEE Trans. on Visualization and Computer Graphics*, 18(3):407–420, 2012.

[41] A. Saad, T. Möller, and G. Hamarneh. Probexplorer: Uncertainty-guided exploration and editing of probabilistic medical image segmentation. *Computer Graphics Forum*, 29(3):1113–1122, 2010.

[42] T. Schultz and G. Kindlmann. Superquadric glyphs for symmetric second-order tensors. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1595–1604, 2010.

[43] T. Schultz, H. Theisel, and H.-P. Seidel. Segmentation of DT-MRI anisotropy isosurfaces. In K. Museth, T. Möller, and A. Ynnerman, editors, *Proc. Eurographics/IEEE-VGTC Symposium on Visualization (EuroVis)*, pages 187–194, 2007.

[44] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[45] C. T. Silva, J. Freire, and S. P. Callahan. Provenance for visualizations: Reproducibility and beyond. *Computing in Science & Engineering*, 9(5):82–89, 2007.

[46] G. W. Stewart. A krylov–schur algorithm for large eigenproblems. *SIAM J. on Matrix Analysis and Applications*, 23(3):601–614, 2001.

[47] A. Tatu, F. Maaß, I. Färber, E. Bertini, T. Schreck, T. Seidl, and D. Keim. Subspace search and visualization to make sense of alternative clusterings in high-dimensional data. In *Proc. IEEE Symp. on Visual Analytics Science and Technology (VAST)*, pages 63–72, 2012.

[48] T. Torsney-Weir, A. Saad, T. Möller, H.-C. Hege, B. Weber, and J.-M. Verbavatz. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Trans. on Visualization and Computer Graphics*, 17(12):1892–1901, 2011.

[49] F.-Y. Tzeng and K.-L. Ma. Opening the black box – data driven visualization of neural networks. In C. Silva, E. Gröller, and H. Rushmeier, editors, *Proc. IEEE Visualization*, pages 383–390, 2005.

[50] J. Vollmer, R. Mencl, and H. Müller. Improved laplacian smoothing of noisy surface meshes. *Computer Graphics Forum*, 18(3):131–138, 1999.

[51] U. von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[52] M. R. Wiegell, D. Tuch, H. B. Larsson, and V. J. Wedeen. Automatic segmentation of thalamic nuclei from diffusion tensor magnetic resonance imaging. *NeuroImage*, 19:391–401, 2003.

[53] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *Proc. ACM Conf. Knowledge Discovery and Data Mining (SIGKDD)*, pages 907–916, 2009.