Machines That Learn to Segment Images: a Crucial Technology for Connectomics

Viren Jain, PhD; H. Sebastian Seung, PhD; and Srinivas C. Turaga, PhD

Howard Hughes Medical Institute
Brain and Cognitive Sciences
Department of Massachusetts Institute of Technology
Cambridge, Massachusetts
Introduction

Connections between neurons can be found by checking whether synapses exist at points of contact, which in turn is determined by neural shapes. Finding these shapes is a special case of image segmentation, which is laborious for humans and would ideally be performed by computers. New metrics properly quantify the performance of a computer algorithm using its disagreement with "true" segmentations of example images. New machine learning methods search for segmentation algorithms that minimize such metrics. These advances have reduced computer errors dramatically. It should now be faster for a human to correct the remaining errors than to segment an image manually. Further reductions in human effort are expected and crucial for finding connectomes more complex than that of *Caenorhabditis elegans*.

Imaging Technology and Its Discontents

Imaging technologies have influenced biology and neuroscience profoundly, starting from the cell theory and the neuron doctrine. Today’s golden age of fluorescent probes has renewed the belief that innovations in microscopy lead to new discoveries. But much of the excitement over imaging overlooks an important technological gap: scientists need not only machines for making images, but also machines for seeing them.

With today’s automated imaging systems, it is common to generate and archive torrents of data. For some experiments, the greatest barrier is no longer acquiring the images, but rather the labor required to analyze them. Ideally, computers would be made smart enough to analyze images with little or no human assistance. This is easier said than done: it involves fundamental problems that have eluded solution by researchers in artificial intelligence for half a century.

One of these problems is image segmentation — the partitioning of an image into sets of pixels (segments) corresponding to distinct objects. For example, a digital camera user might like to segment an image of a room into people, pieces of furniture, and other household objects. A radiologist may need the shapes and sizes of organs in an MRI or CT scan. A biologist may want to find the cells in a fluorescence image from a microscope. Engineers have tried to make computers perform all of these tasks, but computers still make many more errors than humans.

Recently, there has been progress in answering two basic questions about image segmentation:

1) Given two different segmentations of the same image, how can the amount of disagreement between them be quantified?

2) Given a space of segmentation algorithms, how can a computer be used to search for a good algorithm?

In the past few years, the first question has been addressed by the introduction of metrics that mathematically formalize our intuitive notions of “good” segmentation. These metrics penalize topological disagreements between segmentations and are less sensitive to small differences in boundary locations (Unnikrishnan et al., 2007; Jain et al., 2010). The new metrics are significant, because they can be applied to quantify the performance of a computer algorithm by measuring its disagreement with “true” segmentations of a set of example images (generally provided by humans). Good metrics are absolutely essential for progress in research. Without them, it is not even possible to tell whether progress is being made.

The second question has been answered by formulating the search as an optimization. That is, one can use a computer to search for an algorithm that minimizes disagreement with the true segmentations, as measured by the new metrics (Turaga et al., 2009; Jain et al., 2010). Such automated search is called “machine learning” from examples. It is distinct from the conventional approach, in which a human directly designs a good algorithm using intuition and understanding. Many still adhere to the conventional approach, which has produced a huge number of papers over decades of research (a Google Scholar query for the phrase “image segmentation” yields more than 200,000 references), but empirical results have shown that machine learning produces superior accuracy (UC Berkeley College of Engineering, 2009). The utility of machine learning has already become accepted for other computer vision tasks, such as object recognition (Forsyth and Ponce, 2002), and we expect that it will become standard for image segmentation as well. The above innovations are quite general, but applications to just two image domains will be discussed in this chapter.

Research into segmenting so-called natural images, or photographs of ordinary scenes, began in the late 1960s (Rosenfeld, 1969). Research into segmenting serial electron microscopic (serial EM) images of...
neurons began in the 1970s (Sobel et al., 1980) and has largely applied algorithms or ideas that were originally developed for natural images. In the past few years, however, this niche area has given rise to innovations that have yet to be applied to natural images. Note that natural images are two-dimensional (2D), while serial EM images are three-dimensional (3D). The ideas discussed in this chapter are applicable to both types of images, and more generally to arbitrary dimensionality.

Serial EM produces a 3D image one slice at a time, generating a virtual stack of 2D images (Harris et al., 2006). By segmenting serial EM images of neurons, one can find their shapes, including the trajectories of their axons and dendrites. The shapes of neurons are important because they determine whether neurons contact each other. By checking all contact points for synapses, it is possible to map all the connections between neurons in order to find a connectome (Sporns et al., 2005). This process was carried out for the nematode Caenorhabditis elegans in the 1970s and 1980s (White et al., 1986). Although the C. elegans connectome contains just 7000 connections between 300 neurons, it took more than a decade to find. Most of the time was spent on image analysis, which was performed without the aid of computers.

Recent advances in serial EM (Denk et al., 2004; Briggman and Denk, 2006; Hayworth et al., 2006; Smith, 2007) have revived interest in finding connectomes. These improved methods promise to produce images of larger volumes of brain tissue. (The focused ion beam/scanning electron microscopy [FIB/SEM] method also produces high-resolution images, but it is not yet clear whether it can be scaled up to larger volumes.) A cubic millimeter is estimated to require up to hundreds of thousands of person-years of human effort to segment manually (Helmstaedter et al., 2008). From such numbers, it is obvious that the need for automated segmentation has become even more acute.

This review focuses only on image segmentation. We will not address the automation of synapse detection, because this important problem is little studied so far. Arguments for the importance of connectomes to neuroscience can be found elsewhere (Lichtman and Sanes, 2008; Seung, 2009). Finally, we do not address reconstruction of isolated neurons from the sparse images generated by light microscopy. In the limit of well-isolated neurons, this is not a problem of segmenting multiple objects, but rather of finding the best description of a single object as a tree.

The Segmentation Problem

The following two definitions of the segmentation problem are equivalent:

Definition 1. Segmentation as partitioning. Partition the image into sets of pixels called segments, which correspond to distinct objects.

Definition 2. Segmentation as an equivalence relation. Decide whether each pair of pixels belongs to the same object or different objects.

Definition 1 is more intuitive to most people, while Definition 2 is useful for some of the formalism described below. The definitions are equivalent because of the mathematical fact that any partitioning corresponds to an equivalence relation.

Region coloring

It is common to display the result of segmenting an image using a region coloring, which assigns colors to the pixels of an image, such that different colors correspond to different objects. Example colorings are shown in Figure 1. A coloring may reserve a special color for pixels that do not belong to any object. These pixels belong to boundaries between objects or to the background. It is trivial to turn a coloring into a partitioning: two pixels belong to the same segment if and only if they have the same color. A coloring is a nonunique representation of a segmentation, since any permutation of colors leads to the same partitioning. A coloring is also a nonlocal representation, in the sense that the coloring of different pixels cannot be done independently.

Boundary detection

As their first stage, many segmentation algorithms perform the following computation:

Boundary detection. Decide whether each pixel belongs to a boundary between objects.

The result of this computation is a “boundary labeling”: a black-and-white image in which white pixels correspond to boundaries, and black pixels correspond to interiors of objects (Fig. 1).

A second stage transforms the boundary labeling into a segmentation (as in Definition 2) by using connectedness as an equivalence relation between pixels. Two interior pixels are said to be connected if there exists a path between them that traverses only interior pixels in the boundary labeling. Connected sets of interior pixels (connected components) correspond to segments of a partitioning or coloring.
Figure 1. Segmentation of various types of images. a, Natural image from the Berkeley Segmentation Dataset (Martin et al., 2004); scanning EM image of mouse cortical neurons (image courtesy of the Lichtman lab, annotation by Daniel Berger); and fluorescent image of actin-labeled Drosophila melanogaster cells (Jones et al., 2005). In all cases, segmentation has been provided by human annotation. b, Cross-sections of a dendrite (red) and axon (green) in an EM image of hippocampus, colored by hand using ITK-SNAP software, and 3D reconstructions of the axon making a synapse onto a spine of the dendrite, assembled from EM images of a series of 30 nm slices (image courtesy of the Lichtman lab, annotation by Daniel Berger).
stage of partitioning is inevitably nonlocal because it involves finding out whether pairs of pixels are connected, and these pixels may be distant from each other. And, even if the two pixels are nearby, the path connecting them might travel arbitrarily far away.

As shown in Figure 1, the first stage of boundary detection would ideally be nonlocal also, because there are difficult locations in the image where boundaries cannot be accurately detected without contextual information from distant pixels. However, for most locations, nearby pixels are sufficient for making the correct decision. Therefore, many boundary detection algorithms consider only local information. This constraint limits accuracy, but it also improves speed—a practical compromise. Local boundary detectors generally look for abrupt changes in various properties such as intensity, color, and texture (Sobel and Feldman, 1968; Marr and Hildreth, 1980; Sobel et al., 1980). Such algorithms are said to be gradient-based, because the abrupt changes are found by thresholding some kind of spatial derivative.

Because local boundary detectors are quite inaccurate (Fig. 2), their output is often fed to a subsequent stage of computation that is supposed to use contextual information to correct the errors. The Canny edge detector was a simple version of this idea (Canny, 1986). More sophisticated methods included relaxation labeling (Parent and Zucker, 1989); nonlinear diffusion (Perona and Malik, 1990); Markov random fields (Geman and Geman, 1986; Li, 1994), which can sometimes be optimized efficiently using graph cuts (Boykov et al., 2001); active contours (Kass et al., 1988); and level sets (Osher and Fedkiw, 2001; Angelini et al., 2005).

These methods all involve interesting mathematics, and their proponents like to focus on the differences between them. We prefer to regard the methods as more similar than different. All define the dynamics (or an optimization) of auxiliary variables associated with the pixels. Each variable is updated depending on a linear combination of variables from neighboring pixels, as well as some kind of nonlinear operation. Iteration of the dynamics propagates information over long distances.

Also applied to EM images of neurons are nonlinear diffusion (Tasdizen et al., 2005), Markov random fields optimized by graph cuts (Yang and Choe, 2009; Kaynig et al., 2010), level sets (Macke et al., 2008; Vazquez-Reina et al., 2009; Jeong et al., 2009, 2010), and active contours (Carlholm et al., 1994; Vazquez et al., 1998; Bertalmio et al., 2000; Jurrus et al., 2009), mostly during the last decade.

In practice, the above algorithms generate analog values rather than binary labels. These values are sometimes interpretable as the probability that a pixel belongs to a boundary. The analog boundary labeling, or boundary map, can be thresholded to produce a binary labeling, which is then used to find connected components.

In EM images, even a low rate of missed boundary pixels in the boundary map can result in undersegmentation by the connected components procedure. In practice, this is prevented by using a low threshold for boundary detection, resulting in a high false-positive rate but a low false-negative rate, and thus, fewer mergers (Andres et al., 2008).

The regular watershed algorithm is an alternative approach to creating a segmentation from a boundary map (Vincent and Soille, 1991). This approach is distinct from connected components but closely related. The watershed algorithm tends to oversegment the image, producing many more segments than objects. This is because there is a watershed domain for each local minimum of the boundary map, and local minima are typically very numerous. Therefore, watershed is generally augmented by schemes for damping local minima or merging watershed domains to form larger segments.

Affinity graph labeling
Boundary detection is not the only possible first stage for a segmentation algorithm. An alternative is to label the edges of an affinity graph, which consist of nodes corresponding to image pixels.

Affinity graph labeling. Label each affinity graph edge to indicate whether its pixels belong to the same or different objects.

Each edge label is called an affinity. As with a boundary labeling, a second stage of computation is required to transform the affinity graph into a segmentation. This second stage has two goals:

• First, the resolution of inconsistencies. Suppose that an affinity graph is fully connected, containing all possible edges between nodes. Then labeling its edges would seem to specify an equivalence relation between pixels, as in Definition 2 of segmentation. But the edge labels may violate the property of transitivity, and hence be inconsistent with an equivalence relation. These inconsistencies must be resolved to produce a segmentation; and
Second, to supply missing information. If the affinity graph is only partially connected, then it only partially specifies an equivalence relation. Therefore, the second stage of computation must decide about the missing edges of the graph, as well as resolve inconsistencies.

Both of these goals can be accomplished by defining connectedness in the graph to be an equivalence relation. Two nodes are said to be connected in the graph if there exists a path between them traversing only edges with affinity equal to one. The affinity graph is partitioned into connected sets of nodes by the second stage of computation.

Various types of connectivity have been used in affinity graphs. One type of partial connectivity contains edges only between adjacent or nearest neighbor (NN) pixels. Such a NN graph has been used for EM images (Turaga et al., 2010) as well as natural images (Fowlkes et al., 2003). An NN affinity graph is quite similar to a boundary labeling. (Note, however, that an affinity graph has the opposite sign convention as a boundary labeling, which can be confusing.) The only difference is that boundaries are located at the midpoints between pairs of adjacent voxels rather than at the voxels themselves. Adjacent voxels can belong to different objects in an NN affinity graph, but not in a boundary labeling.
In a boundary labeling, boundaries are represented by voxels that do not belong to any cell but rather to extracellular or “outside” space. An NN affinity graph does not need to assign voxels to boundaries, since it uses edges to represent boundaries. However, it has the option of assigning voxels to extracellular space, in which case they end up being disconnected from each other and from all cells. Thus, an NN affinity graph is more powerful than a boundary labeling, in the sense that it can represent more partitionings of the image. Of course, this power is achieved by including more information: there are more edges in a NN affinity graph than voxels in a boundary labeling.

NN affinity graphing can be advantageous for representing segmentations of EM images with limited spatial resolution. Where neurites become very thin, there may not be enough voxels to represent both the interiors of the neurites as well as the boundaries between them. Here, the true interpretation of the image may be ambiguous based on local information, but can become unambiguous when more context is included. This problem is solved by representing boundaries using edges between voxels. Also, if the 3D image is composed of aligned 2D images of physical slices, errors in alignment can cause voxels of two different cells to end up adjacent to each other across two slices, with no extracellular space between them. Another way of dealing with this problem is to compute a super-sampled output image that has higher resolution than the input image (Jain et al., 2007).

An affinity graph can also contain edges between pairs of voxels that are not NNs; in this situation, it is less like a boundary labeling. An affinity graph can represent a segment that consists of a set of voxels that is disconnected in the image, as long as the set is connected in the graph. This is useful for EM images with limited spatial resolution, as thin neurites can “break” when they become less than one voxel in diameter. Although such neurites are disconnected in the image, they can be connected by long-range edges in the affinity graph. Similarly, if the 3D image is composed of aligned 2D images of physical slices, errors in alignment can cause a thin neurite to end up disconnected in the image. (In natural images, partial occlusion can split a background object into regions that are disconnected in the image. Such an object can still be connected in an affinity graph with long-range edges.)

All these advantages of the affinity graph can be summarized in a single bottom line: connectedness is based on the definition of adjacency, and this definition is more flexible in an affinity graph than in a boundary labeling.

As with boundary detection, achieving the highest accuracy at labeling an edge of an affinity graph would require contextual information from distant pixels in the image. In practice, however, local algorithms are often used for the sake of speed. For NN edges, the definition of local is basically the same as it was for boundary detection. For labeling long-range edges, an algorithm is said to be local if its output depends only on the two image patches surrounding the two pixels connected by the edge. The affinity is generally computed using some measure of similarity of the two image patches, based on properties such as intensity, color, and texture (Fowlkes et al., 2003).

As with boundary labelings, it is common for algorithms to generate analog values rather than binary labels for the affinity graph edges. These values are sometimes interpretable as the probability that an edge connects two pixels that belong to the same object. These analog values can be thresholded to produce a binary edge labeling, which is then used to find connected components.

In addition to connected components, many other algorithms have been proposed for partitioning an affinity graph. These are supposed to correct the errors and inconsistencies that local computation of the affinity graph labeling produces. Spectral methods have been applied to produce a hierarchical partition of the affinity graph (Shi and Malik, 2000). An analog of the watershed algorithm can be defined for graphs through the minimum spanning tree (Felzenswalb and Huttenlocher, 2004; Cousty et al., 2008).

**Manual creation of segmentation datasets**

Two broad classes of segmentation algorithms were defined above: those that involve boundary detection and those that involve affinity graph labeling. Rather than describing more classes of algorithms, we move now to a different subject — that of evaluating performance. Surprisingly, this issue was not confronted seriously until the 2000s. Previously, researchers had evaluated algorithms subjectively, by inspecting performance on a few images. Without objective and quantitative means of evaluation, it was difficult to tell which algorithms were better.

Usually it is easy to evaluate performance in computer science. Many computational tasks, like multiplying numbers or inverting matrices, have simple and precise
mathematical specifications, so that it is straightforward to measure speed and accuracy. But no explicit specification exists for image segmentation and other tasks in computer vision. There is no way to evaluate performance without resorting to empirical means.

This was first done in a systematic way by the introduction of the Berkeley Segmentation Dataset (Martin et al., 2004). Its creators collected natural images and employed humans to segment them. Both images and segmentations were made publicly available. Researchers could quantify and compare the performance of their computer algorithms by measuring disagreement with the human segmentations using a common dataset. More recently, researchers have created similar datasets that contain EM images along with segmentations (Jain et al., 2007; Turaga et al., 2009, 2010; Cardona, 2010; Jain et al., 2010). So far, one of them is publicly available (Cardona, 2010).

To manually segment a 2D image, a human can use standard computer software for drawing or painting. In boundary tracing, the human draws contours at the boundaries of objects. In region coloring, the human paints the interiors of objects with different colors. These methods can be extended to 3D images by allowing the user to annotate 2D slices.

The software packages RECONSTRUCT (Fiala, 2005), TrakEM2 (Cardona, 2006), and KLEE (M. Helmstaedter, personal communication) are capable of boundary tracing of EM images. As illustrated in Figure 1, ITK-SNAP (Yushkevich et al., 2006) can be used for region coloring of biomedical images.

For segmenting neurons, there is an alternative that is faster and captures most of the shape information. A human can draw a line along the axis of a neurite, with the capability of adding branch points. This functionality is implemented by the software packages TrakEM2, Elegance (S. Emmons, unpublished data), and KNOSSOS (M. Helmstaedter, unpublished data). Such skeleton tracing is >10 times faster than full segmentation (Helmstaedter et al., 2008). SSECRET (Jeong et al., 2010) and Piet (J. Lichtman, unpublished data) allow the dropping of “bread crumbs” in order to summarize neurons as sets of points. This method is a bit less informative than skeletons, which include the lines connecting the points.

Neuro3D (Macke et al., 2008) and NeuroTrace (Jeong et al., 2009, 2010) are software packages for semiautomated tracing of neurites that combine active contours or level sets with a graphical user interface. For each 2D image slice, the computer suggests a contour for the boundary of the neurite. The human user either accepts this contour or corrects it. NeuroTrace utilizes parallel computation with graphics processing units to speed up the level set computation.

Metrics of segmentation performance
The introduction of common segmentation datasets is essential for allowing researchers to properly quantify and compare the performance of their computer algorithms. But datasets alone are not enough. It turns out that defining a proper metric for measuring disagreement between segmentations is a nontrivial problem to which good solutions have been proposed only recently.

In general, a metric can be used to compare any pair of segmentations. Most commonly, one of the segmentations comes from a computer and the other from a human. If the human segmentation is regarded as the “truth,” then the metric measures the error of the computer. Therefore, we will often use the term “error” interchangeably with “metric.” Metrics can also be used to compare two human segmentations. Consistency of human segmentations is an important indication of whether it makes sense to regard them as the “truth” (Martin et al., 2004; Jain et al., 2007; Mishchenko, 2009).

Outlook
New performance metrics, as well as machine learning methods based on them, are transforming research on image segmentation. These innovations have largely been driven by the goal of segmenting serial EM images of neurons. One might ask why this niche application has played a disproportionately important role. One reason is that the shapes of neurons are highly complex, making accurate segmentation extremely difficult and forcing researchers to try new ideas. A second reason is that a serial EM image actually possesses a “true” segmentation, which neuroscientists really want to know. In contrast, the notion of a segmentation is not completely well defined for natural images, as evidenced by the fact that human segmentations of images in the Berkeley Segmentation Dataset often disagree substantially. Researchers may not have a strong incentive to achieve very low error rates, since this may be fundamentally impossible anyway.

As described above, machine learning approaches have created new algorithms for segmenting...
EM images with outstanding accuracy relative to conventional algorithms. For the first time, semiautomated segmentation is becoming faster than purely manual segmentation (though more precise quantification of this claim is needed). This is an important milestone, demonstrating the utility of computerized image segmentation, but it is just the beginning. It is important to further reduce human effort that is consumed by semiautomated segmentation. How can computer accuracy be improved even further? And more fundamentally, why should we believe that further gains are possible at all?

What are the challenges involved in increasing the field of view? First, the computation may become slower, owing to the demands of processing more information. In 3D, doubling the length of the field of view increases the volume by almost an order of magnitude. Second, a larger field of view will provide not simply more contextual information, but information of a different type. Utilizing it may require a computation that is quite different. Therefore, it is unrealistic to proceed by simply scaling up a single monolithic computation to an arbitrarily large field of view. (It is possible to write down a very simple Markov random field model that utilizes an unlimited field of view; however, this model would use a single mechanism to propagate information over both short and long distances and therefore could not cope with multiple types of contextual information at different length scales.) One possibility is to modify existing boundary detection techniques to perform multiscale computation (Ren, 2008) by combining separate computations at multiple spatial resolutions. If boundary detection is followed by another stage of automating splitting and merging of supervoxels, this could potentially provide an efficient and powerful means of dealing with added context. Each supervoxel should be represented by a descriptor more compact than its raw voxels. Ideally, future research will yield shape descriptors that allow fast and accurate decisions to be made about split and merge operations.

Acknowledgments
This paper was published in unabridged format in Current Opinion in Neurobiology (2010;20:653-66). Please refer to the published paper for complete references and recommended reading list, and for more information on machine learning from examples, semiautomated segmentation strategies, a sample affinity graph, and images of segmentation quality.

References


NOTES


© 2011 Seung