A new method for more efficient seismic image segmentation

Adam Halpert

ABSTRACT

Seismic images of the subsurface are often very large and tedious to interpret manually; as such, automatic segmentation algorithms can be highly useful for tasks such as locating large, irregularly-shaped salt bodies within the images. However, seismic images present unique challenges for image segmentation algorithms. Here, a new segmentation algorithm using a "pairwise region comparison" strategy is implemented and tested on seismic images. Numerous modifications to the original algorithm are necessary to make it appropriate for use with seismic data, including: (1) changes to the nature of the input data, (2) the way in which the graph is constructed, and (3) the formula for calculating edge weights. Initial results, including a preliminary 3D implementation, indicate that the new method compares very favorably with an existing implementation of the eigenvector-based normalized cuts approach, both in terms of accuracy and efficiency.

INTRODUCTION

A proper salt interpretation is one necessary component of any imaging project where salt bodies play a prominent role in the subsurface geology. Because of the sharp velocity contrast between salt and nearly any other material, inaccurate placement of salt boundaries has a disproportionate effect on the accuracy of the resulting velocity model. Such errors can have damaging imaging and engineering consequences. Unfortunately, interpreting salt boundaries is not only crucial, but also extremely tedious and time-consuming when undertaken manually. Thus, while some degree of automation would be ideal for salt picking, any such method must be highly accurate as well as efficient.

One approach to implementing automatic salt-picking is to use graph-based image segmentation. In this method, each pixel in a seismic image is treated as a node or vertex in a graph; then edges are constructed between specific pixels and weighted according to some property. Image segments are created by partitioning the graph – here, a partition is a salt boundary.

In this paper, I will review a previous effort to apply a graph-based image segmentation algorithm to seismic data, and then introduce a new, more efficient method using an algorithm from Felzenszwalb and Huttenlocher (2004). Since seismic images

are very different from more "conventional" images (such as photographs) for which this newer algorithm was designed, I will detail several modifications necessary for the algorithm to be useful for seismic images. Finally, I demonstrate the results of applying the new method to the examples seen in Figures 1(a) and 1(b), and (for the field data example) compare these results with those obtained using the previous algorithm.

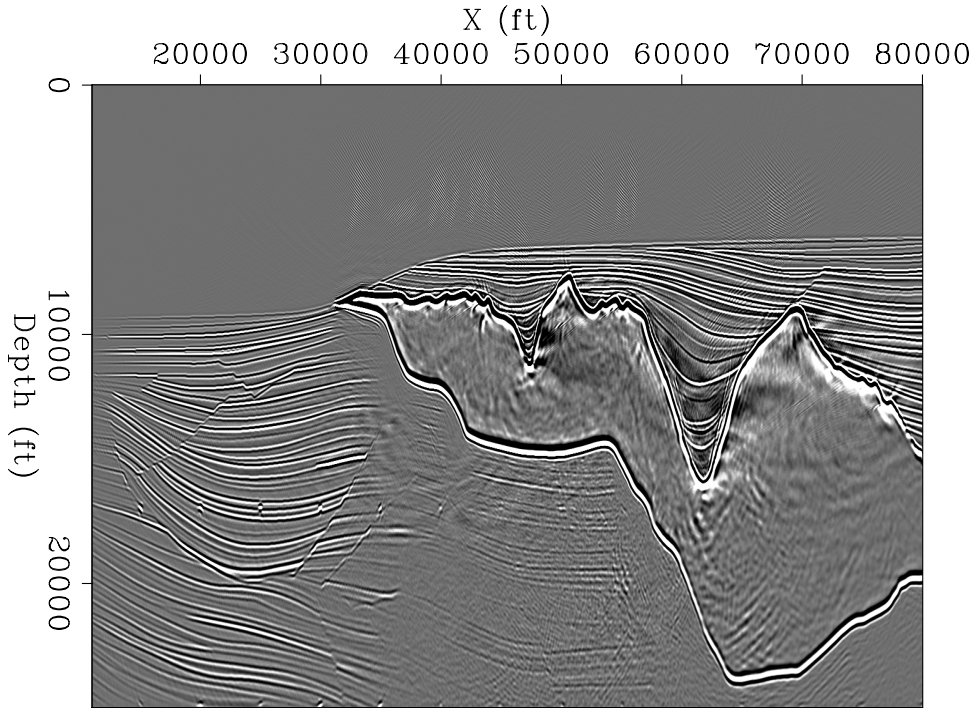
BACKGROUND

As mentioned previously, graph-based segmentation methods are popular for automating salt interpretation. More specifically, the eigenvector-based Normalized Cuts Image Segmentation (NCIS) algorithm (Shi and Malik, 2000) has attracted a great deal of interest because of its capability to capture global aspects of the image, rather than just track local features. Local feature trackers are ubiquitous in seismic interpretation software (for example, tracking reflections between manually-picked "seed" points), but often struggle when they encounter situations such as a discontinuous boundary, or one which varies in intensity. Such situations are extremely common in seismic data, especially along the boundary between salt bodies and other types of rock. Recent research has involved implementing the NCIS algorithm for the purpose of tracking these salt boundaries [e.g., Lomask et al. (2007); Lomask (2007); Halpert et al. (2009)]. Results of this line of research are encouraging; however, there are significant limitations, most notably computational. The NCIS algorithm calls for an edge weight matrix of size n^2, where n is the number of pixels in the image; this matrix quickly grows very large, especially for 3D surveys. Computationally, calculation of eigenvectors for such a large matrix is an extremely demanding task. As such, this method is limited to relatively small images; alternatively, we can restrict the computational domain to a specific region around a previously interpreted boundary. However, this means the method is of limited utility if there is no "best guess" model available, or if the accuracy of that model is in question.

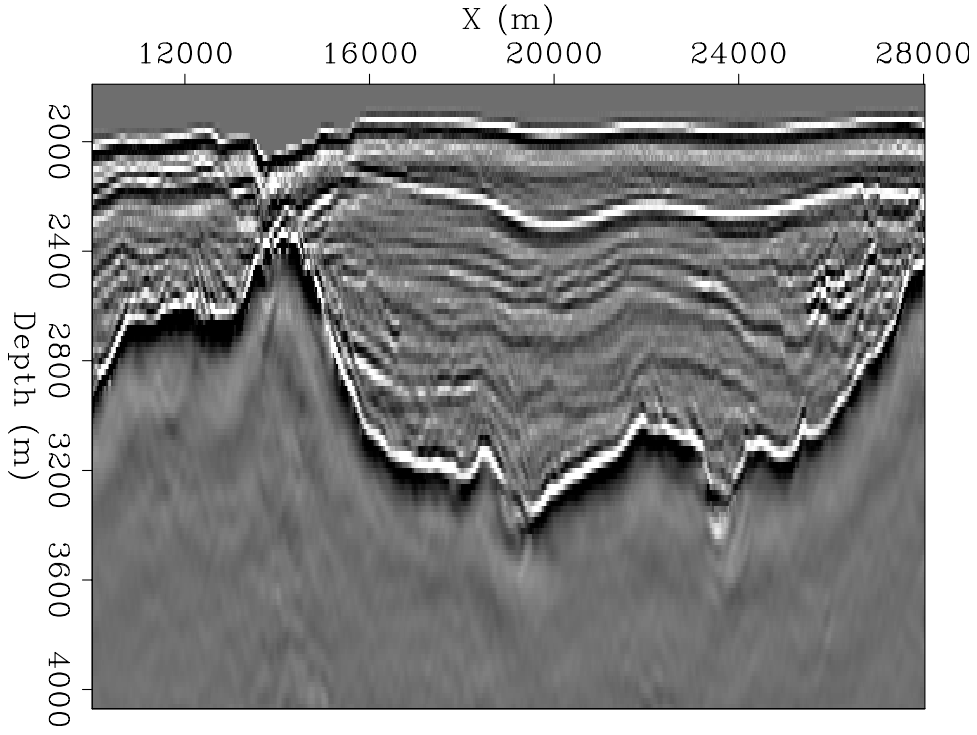
Thus, a more efficient global segmentation scheme that can include the entire image in the computational domain would be a very useful tool for interpretation of seismic images. One candidate for such a scheme is the algorithm from Felzenszwalb and Huttenlocher (2004), who write:

> "Our algorithm is unique, in that it is both highly efficient and yet captures non-local properties of images."

These two features are crucial for the task of seismic image segmentation. The algorithm is designed to run in $O(n \log n)$ time, where n is the number of pixels in the graph; in contrast, other methods such as NCIS require closer to $O(n^2)$ time to run. This represents a significant cost savings, especially for very large 3D seismic datasets that are becoming increasingly common.



(a)



(b)

Figure 1: A perfect-velocity migration of the Sigsbee synthetic model (a), and a field seismic image (b) that will be used as examples throughout this paper. [**ER**]

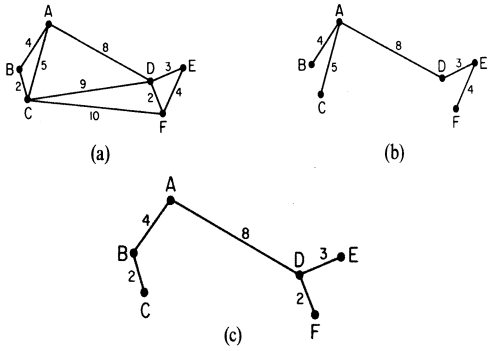


Figure 2: Modified from Zahn (1971). A graph with weighted edges (a); a spanning tree of that graph (b); and the minimum spanning tree of the graph (c). [**NR**]

The algorithm proposed by Felzenszwalb and Huttenlocher (2004) relies heavily on the concept of the "Minimum Spanning Tree" [see Zahn (1971)]. A graph's edges may be weighted using a measure of dissimilarity between vertex pairs; a connected graph is defined as one in which all such edges are assigned a weight value. If a spanning tree is a connected graph which connects all vertices of the graph without forming a circuit, the minimum spanning tree (MST) of a graph is the spanning tree with the minimum sum of edge weights (see Figure 2). In Zahn (1971), partitioning of a graph was achieved simply by cutting through edges with large weights. However, this approach is inadequate for images with coherent regions that are nonetheless highly heterogeneous (for example, consider the heterogeneous nature of the intensity values within the salt bodies in the examples above). However, the MST concept allows Felzenszwalb and Huttenlocher (2004) to develop what they term a "pairwise region comparison" predicate. They define the *internal difference* of a region (C) in the graph to be the largest edge weight of the MST of that region:

$$Int(C) = \max_{e \in MST} w(e), \tag{1}$$

where e is a graph edge and $w(e)$ is the edge's weight, defined according to some simple algorithm. When comparing two regions (such as C_1 and C_2), they define the *minimum* internal difference for the two regions to be

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)), \tag{2}$$

where τ is a thresholding function that in a sense determines the scale at which the segmentation problem is approached, and thus indirectly the size of the regions in the final segmentation. Finally, they define the *difference* between the two regions to be the smallest edge weight that connects them:

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2} w((v_i, v_j)), \tag{3}$$

where v_i and v_j are vertices (or pixels) in the two different regions. When determining whether these two regions should be considered separate segments of the graph, or merged into a single region, they simply compare the values of $Dif(C_1, C_2)$ and $MInt(C_1, C_2)$. If $Dif(C_1, C_2)$ is greater, the "pairwise comparison predicate" is determined to be true, and the two regions are separated. While this is a relatively simple procedure, it is designed to allow highly heterogeneous regions to be segmented as a single component of an image. Additionally, Felzenszwalb and Huttenlocher (2004) note that their algorithm produces segmentations that are "neither too coarse nor too fine," referring to the global capabilities of the segmentation process.

In the next section, I will provide more detail about the algorithm itself, and explain the modifications necessary to make it suitable for use with seismic images.

APPROACH

The goal of this paper is to apply the algorithm of Felzenszwalb and Huttenlocher (2004), introduced above, to seismic data. Publicly-available code from Felzenszwalb (2010) allows for relatively easy implementation for standard images; however, seismic images are very different from photographs or other types of images. The following sections describe the rationale and procedure for modifying and adding additional features to the algorithm in order to apply it to seismic data.

Data input and manipulation

After modifying the original algorithm to work with seismic data rather than integer RGB values, seismic images may be segmented according to the same rules used by Felzenszwalb and Huttenlocher (2004) to segment RGB images. The results can be seen in Figure 3(a) for the synthetic seismic image, and Figure 3(b) for the field data image. In these figures, each segment or region is assigned a random color, and the segments are overlain on the seismic image itself for reference. The results from the (relatively) unaltered algorithm are promising, but require improvement. In Figure 3(a), the salt body is fairly well-resolved. However, it appears that the smaller canyon along the left of the salt body has been improperly included, along with another portion adjacent to the salt body at the far right of the image. The segmentation of the field data example in Figure 3(b) is especially poor; however, it is interesting to note that portions of the salt boundary itself have been recognized as segments. This may be related to the nature of the input data, an issue treated in the next section.

Transformation of input data

Seismic data may be thought of as signals with amplitude and phase varying as a function of time (or depth). This could present problems for any segmentation

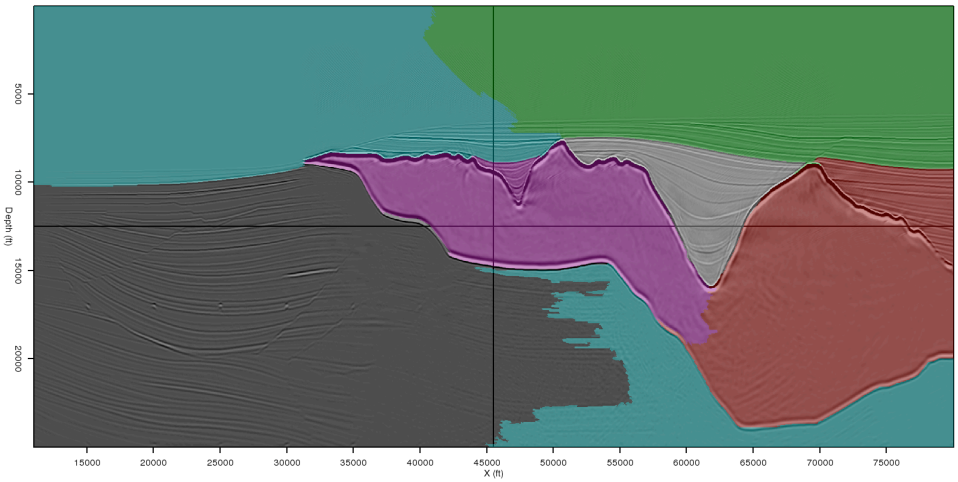
algorithm, and we may see an indication of this in Figure 3(b). At the boundary between the salt body and the surrounding rocks, the seismic waves change phase rapidly; this is common behavior when the waves encounter an interface and reflect back to the surface. As originally written, the algorithm may interpret the area around the boundary as several regions, instead of an interface between just two regions. In this case, the boundary itself becomes its own "region" in several locations. To avoid this situation, we would like the seismic image to be represented as amplitude information only, since this would indicate a single boundary between two regions. As Taner et al. (1979) point out, seismic data may be represented as a complex valued function:

$$A(z)e^{i\phi(z)}, \tag{4}$$

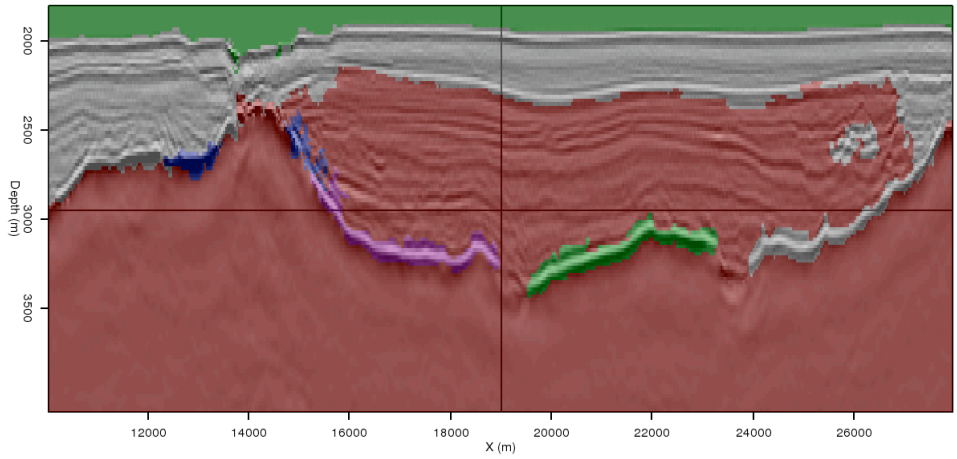
where z can be time or depth. The exponential term in this expression represents the phase information for the seismic data, while the leading term represents the amplitude information. By transforming the data such that the amplitude information is the only information present, the problem described above may be avoided. Figures 4(a) and 4(b) show the result of this process, also known as taking the "amplitude of the envelope" of the data, for the synthetic and field seismic images, respectively. We see that in both instances the phase information is no longer present, and the boundaries delineating the salt bodies are more clearly visible. By using these transformed images as inputs to the segmentation algorithm, it is likely that much of the unwanted behavior seen in the original examples can be avoided.

Creating the graph

The original implementation of the pairwise region comparison algorithm from Felzenszwalb (2010) creates a graph with eight edges per node (pixel). This graph is constructed by looping over every pixel, and performing four calculations at each vertex. The left side of Figure 5 illustrates this process – if the "active" pixel is the one in red, edges are drawn or built to each of the blue pixels. Since every pixel in the image undergoes this process, a form of reciprocity allows for each pixel to be connected to its eight immediate neighbors via edges. While this process allows for the extreme efficiency of the algorithm, the unique and often irregular nature of seismic data does not lend itself well to segmentations using so few edges per vertex or pixel. Instead, a much larger "stencil," shown on the right of Figure 5, has been implemented. Rather than building edges that extend only one pixel in each direction, this stencil creates five edges extending in each horizontal, vertical and diagonal direction from the center pixel. This scheme allows for many more comparisons (40) per pixel, and a far greater amount of information goes into the segmentation algorithm. Near the boundaries of the image, the stencil shrinks to the largest size allowable by the image dimensions. While this approach obviously decreases the efficiency of the algorithm, the increased accuracy seen in the final results appears to make it a worthwhile trade-off. Even with the sharply increased number of edges per node, this algorithm is still far less computationally intensive than the NCIS algorithm from Shi and Malik (2000).



(a)



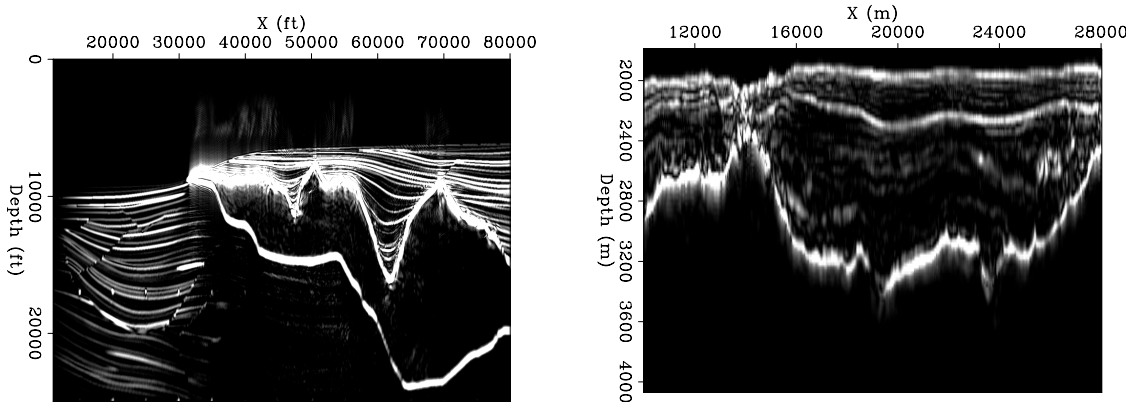
(b)

Figure 3: Segmentation of the example seismic images from Figure 1, using the original algorithm from Felzenszwalb and Huttenlocher (2004). [**NR**]

Calculating edge weights

The previous section details the construction of a graph used for segmentation of seismic images containing 40 edges per pixel. However, perhaps the most important part of any graph-based segmentation scheme is the calculation of weights for each of these edges. It is in this aspect that the algorithm described here differs most from the original implementation provided by Felzenszwalb (2010).

When using the original stencil seen on the left in Figure 5, determining edge weights is relatively straightforward. Since each pair of vertices are at most one pixel apart in the image, we can simply compare the adjacent pixel values and use some expression to determine the likelihood that the two pixels reside in different regions or segments. This process is further simplified by the fact that the original



(a) (b)

Figure 4: Result of calculating the amplitude envelope of the example images seen in Figure 1. These become the input to the new segmentation algorithm. [**ER**]

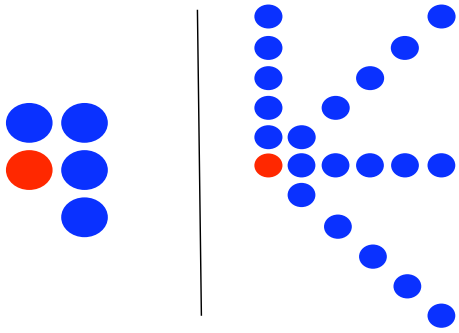


Figure 5: Stencils used for comparing pixel values and assigning edge weights for the graph. At left, the five-point stencil (8 edges per pixel) used in the original implementation from Felzenszwalb and Huttenlocher (2004); at right, a modified 21-point stencil (40 edges per pixel) used for the seismic images. [**NR**]

implementation was designed to segment regions with coherent interiors; that is, even if the interior of a region is relatively chaotic, it is still distinct from the interiors of other regions of the image. However, this is not necessarily the case for seismic images – the interior of a salt body might not be distinct from other areas in the image that lack reflections. In this case a region is defined by its *boundary* rather than the character of its interior. Therefore, we must create a process for calculating edge weights that treats a boundary *between* two vertices as more convincing evidence for the existence of two regions than simply the difference in intensity at the two pixels themselves.

As described in the previous section, the stencil used for determining graph edges forms what are essentially four line "segments" at each vertex – one horizontal, one vertical, and two diagonal. Recall that once the entire graph is constructed, each vertex will actually be connected with eight such segments due to reciprocity of the calculations. Since two vertices that are the endpoints of a graph edge are no longer necessarily adjacent in the image, determining the existence of a boundary between them is no longer straightforward. However, the construction of distinct line segments

extending from each pixel suggests one method of searching for a boundary: existence of a large amplitude value at any point along the line segment between the two pixels comprising the edge is evidence of a boundary. In other words, the intensity values of the two vertex pixels themselves will not be used to determine the edge weight; rather, the largest intensity value of any pixel along the line segment connecting the two endpoint vertices will be used. This strategy follows the approach of Lomask (2007) in his NCIS implementation. Figure 6 illustrates the logic behind this process. If all pixels in a segment are ranked according to intensity, the highest ranked pixel between the two edge vertices will be used for the weight calculation.

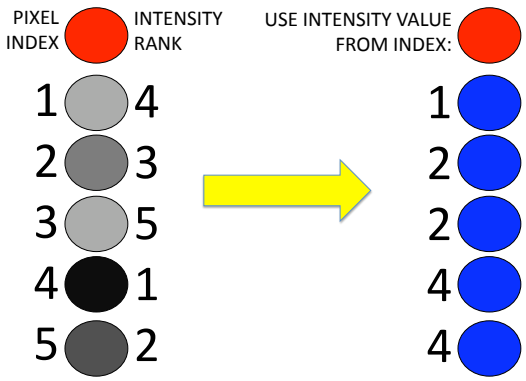


Figure 6: Diagram illustrating the logic behind deciding which pixel intensity value to use when calculating edge weights. Pixel intensities are shown and ranked on the left; the numbers in the right column indicate which intensity value will be used when calculating the edge weight between the pixel in red and the adjacent blue pixel. [**NR**]

This process obviously involves some degree of algorithmic complexity, as it requires sorting and searching the pixel intensity values along each segment. Algorithm 1 illustrates the steps for carrying out the process shown graphically in Figure 6. After creating the edges linking each pixel in a line segment to the "active" pixel, sort the line segment's pixels in decreasing order of pixel intensity. Once this is done, compare the index value of the edge vertex pixel with the intensity-ranked list of pixel indices. To find the highest-intensity pixel value between the two vertices, simply take the value of the first pixel index on the sorted list that is less than the index of the vertex pixel.

Once we have selected the intensity value to use for determining the edge weight, we must still calculate the weight value itself. The original implementation from Felzenszwalb (2010) used a simple Euclidean "distance" between adjacent pixel values. For RGB images, this expression is the equivalent of taking the square root of the sum of the squared differences for each of the three color components at each pixel. However, for the purposes of seismic image segmentation I have found it more appropriate to use an exponential function:

$$w_{ij} = \exp((\max I(\mathbf{p}_{ij}))^2), \tag{5}$$

where $\mathbf{p}_{ij}$ is the vector of all pixels between i and j. Additionally, since the edges in the graph can now be much longer than with the adjacent-pixels-only approach taken in the original implementation, it makes sense to include a distance-weighting term

Algorithm 1 Calculating graph edge weights

```
for each pixel ipix in image do
  create four line segments with five pixels per segment;
  record relative position (path.ind) and intensity (path.val) of each pixel;
  for each line segment do
    sort the segment in decreasing order of pixel intensity;
    for each pixel ix in the segment (nearest to furthest from pixel ipix) do
      for ij = 1...5 do
        if path[ij].ind <= ix then
          calculate edge weight using path[ij].val;
        end if
      end for
    end for
  end for
end for
```

to the edge weight calculation:

$$w_{ij} = \exp((\max I(\mathbf{p}_{ij}))^2) \exp(d_{ij}), \tag{6}$$

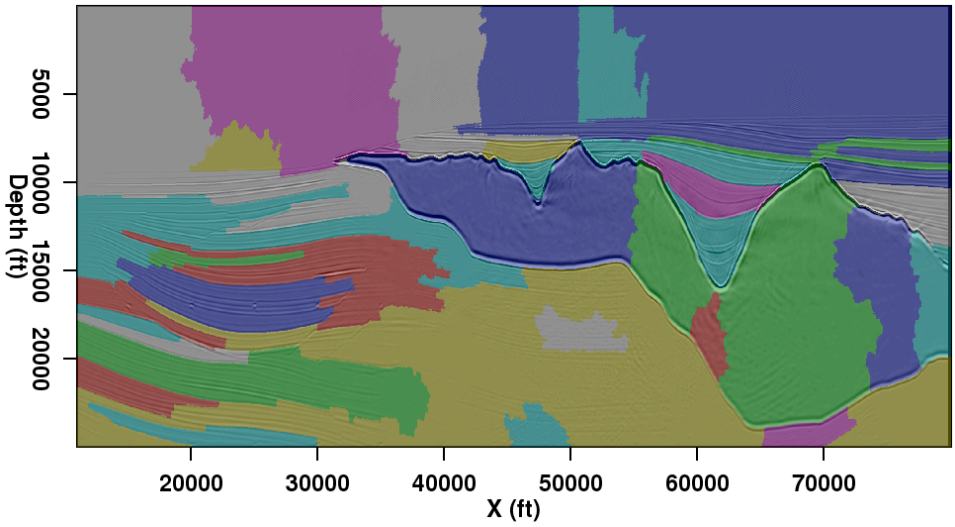
where d_{ij} is simply the Euclidean distance (in samples) between the two pixels.

Once each of the edges is assigned a weight, the segmentation of the image can proceed as described in Felzenszwalb and Huttenlocher (2004). In summary, the process begins with each pixel as its own image segment; then individual pixels, and eventually, groups of pixels, are merged according to the criteria set forth in section 2. Segments can also be merged in post-processing if they are smaller than a "minimum segment size" parameter specified by the user.

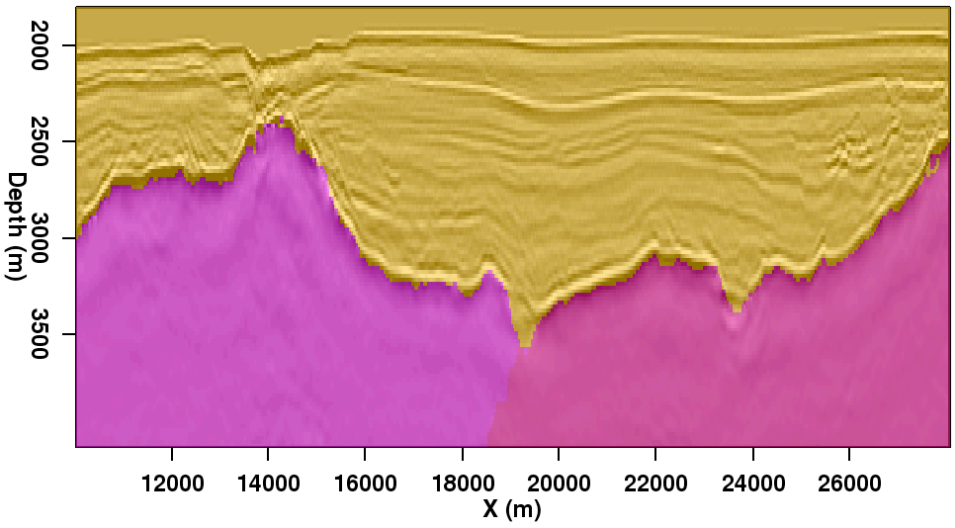
RESULTS

The results of the procedure detailed in the sections above can be seen in Figures 7(a) and 7(b). In Figure 7(a), although the salt body has been segmented into several regions, all of these regions are contained within the salt body and fully conform to its boundaries. This represents an improvement over the original segmentation in Figure 3(a). The improvement is even more dramatic for the field data example (original image in Figure 1(b)). In Figure 7(b), we see that the salt body has been segmented virtually perfectly; no longer do the salt body segments spill over the boundary, nor are parts of the boundary itself treated as individual segments as we saw in the original example.

A major motivation behind this research was to improve on segmentation results using the NCIS method from Shi and Malik (2000), and adapted for use with seismic images by Lomask et al. (2007). The first indication that the newer method does



(a)



(b)

Figure 7: Results of applying the modified segmentation algorithm to the example images from Figure 1. Both (a) and (b) are a significant improvement over the original results in Figures 3(a) and 3(b). [**ER**]

indeed represent a substantial improvement is the fact that the synthetic image from Figure 1(a) is simply *too large* to be segmented on a single processor using the existing NCIS implementation. The necessity of holding a giant, sparse weight matrix in memory and calculating an eigenvector precludes problems of this size from being feasible. The smaller field data example, however, is well-suited for the NCIS algorithm, and will allow us to make a relatively fair comparison. Figure 8(a) shows the eigenvector produced by the NCIS method to segment the image. In this case, the graph partition will occur between the negative (red) and positive (blue) values of the eigenvector. The boundary resulting from this partition is drawn on the image in Figure 8(b). Recall that in order to make the problem more computationally feasible, the computational domain is already limited around a "prior guess" of the boundary. The pairwise region comparison method developed here requires no such limitation, which is another indication of its superiority.

Efficiency comparison

One of the primary means of comparison for the relative effectiveness of these two approaches to image segmentation is the computational efficiency of the method. The following table summarizes the computational expense required to create the examples seen in the paper.

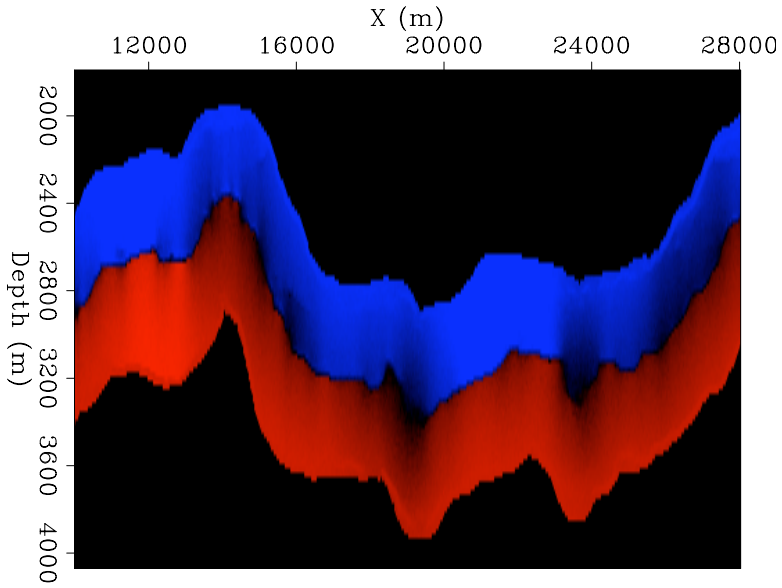
Table 1: Comparison of CPU times for two segmentation methods

		CPU time (s)	
Image type	**Pixels**	NCIS	PRC
Synthetic data	2761000	n/a	31
Field data	55000	156	1

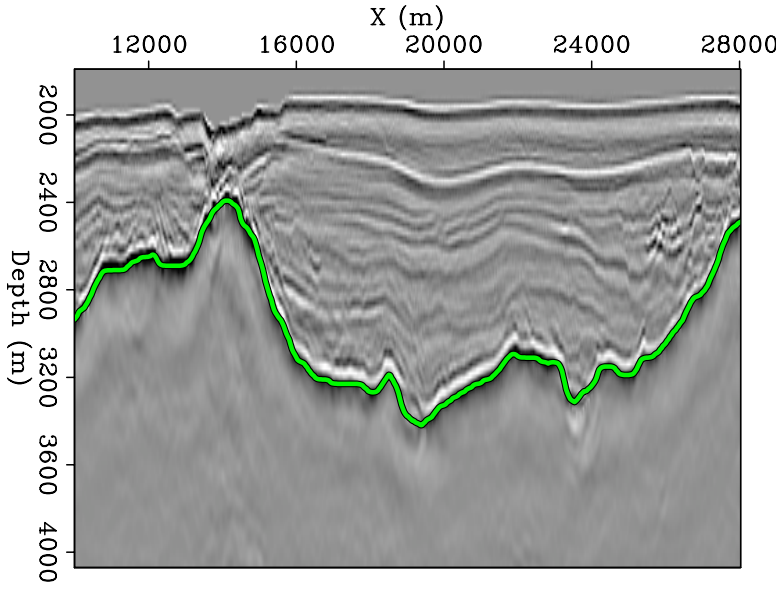
Again, due to memory constraints the existing NCIS implementation is unable to segment an image the size of Figure 1(a). The implementation described here, however, produces an accurate segmentation in 31 seconds; during this time, approximately 55 million edges are created, weighted, and used to segment the graph. The efficiency advantage for the new implementation is quantified using the field data example; in this case, the image is segmented over 150 times faster using the new implementation. These differences are extremely significant and represent a huge savings of time and computational expense, especially for larger problems.

Accuracy comparison

Of course, computational efficiency means little if the resulting segmentation is not accurate. For the synthetic image result in Figure 7(a), we can get a qualitative sense for the accuracy of the new segmentation implementation; while the salt body is



(a)



(b)

Figure 8: Eigenvector (a) used to segment the image in Figure 1(b) according to the NCIS algorithm of Shi and Malik (2000) and adapted for seismic data by Lomask et al. (2007), and the resulting salt boundary (b). [**CR**]

divided into multiple segments, these segments do indeed fit almost exactly inside the boundaries of the salt body. A more direct comparison of the relative accuracy of the NCIS and PRC methods can be obtained via analysis of the field data results. Figure 9 shows both calculated salt boundaries overlain on the image: the NCIS boundary in green, and the PRC boundary in pink. Visually, we can see very little difference between these two results; in many locations, they are almost exactly on top of one another. The most noticeable difference between the two results is near $X = 20000m$, where the PRC boundary dips deeper than the NCIS boundary. Examination of the input image in Figure 1(b) suggests that in this location, an error in the migration velocity model has led to a discontinuity in the boundary image. The new method appears to do a better job of "correcting" the error. This result serves to increase confidence in the viability of this new segmentation scheme as an alternative to the existing NCIS implementation.

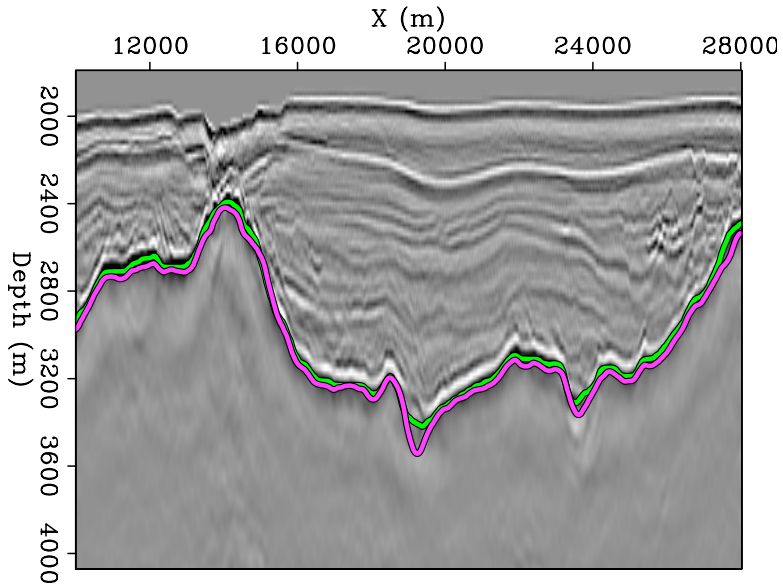


Figure 9: Comparison of the boundaries obtained using the NCIS eigenvector method (green) and the pairwise region comparison method (pink). [**CR**]

3D implementation

The pairwise region comparison strategy described here is easily extendable to three dimensions. While the theory and general approach remain the same as with the two-dimensional case, in three dimensions the stencil used to construct the graph must change. Along with the four co-planar segments extending from each pixel used for the second stencil in Figure 5, a 3D stencil must incorporate four additional segments that extend the graph edges into the extra dimension. Fortunately, the additional information this brings to the segmentation algorithm allows for the stencil's segments to shrink in length; initial tests indicate that the most accurate results are obtained

with segments that are three pixels in length. Thus, the computational impact of extending any algorithm from two to three dimensions is somewhat mitigated in this case.

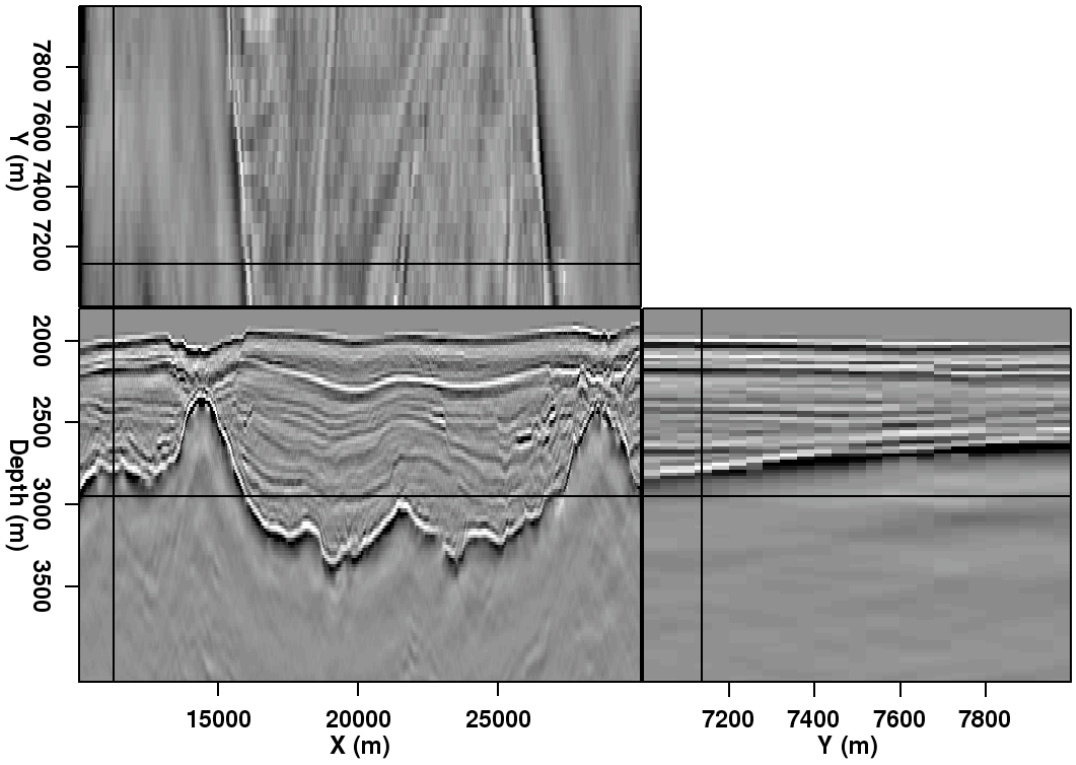
Figures 10(a) and 10(b) show a segmentation example resulting from a preliminary 3D implementation of the new algorithm. While further improvements seem necessary, note that both the salt body (in multiple segments) and the water column have been accurately identified.

CONCLUSIONS

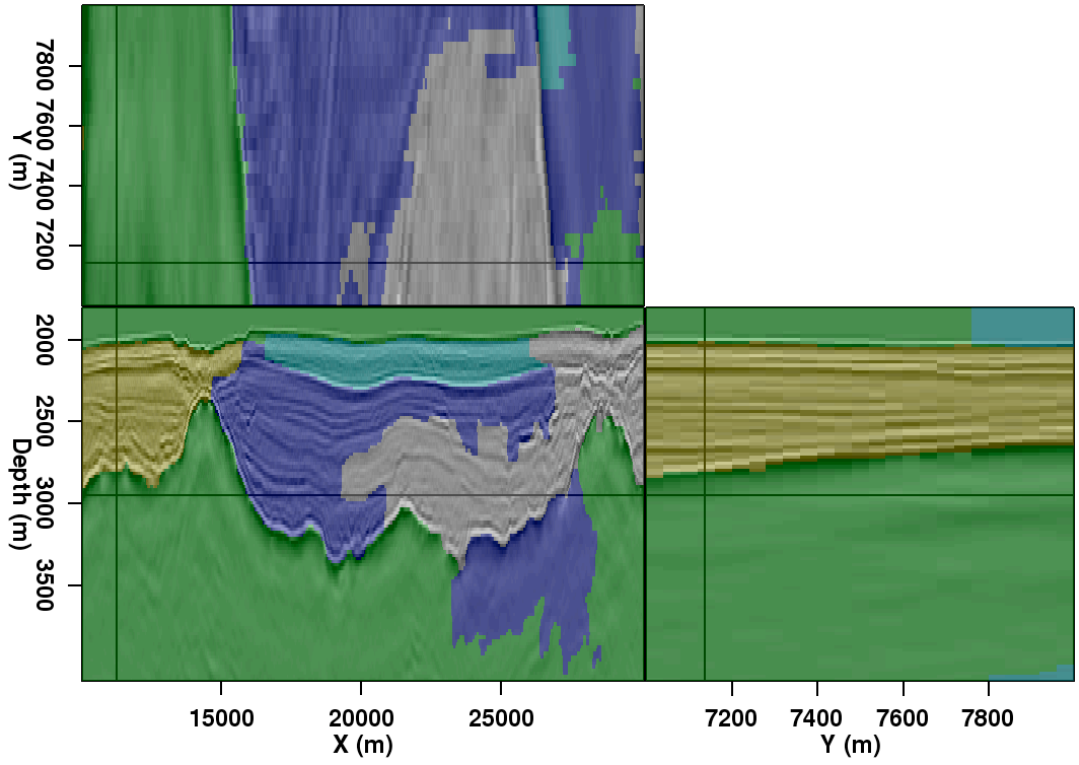
This paper presented an implementation of the pairwise region comparison (PRC) scheme of Felzenszwalb and Huttenlocher (2004) for segmenting seismic images. Numerous modifications were made to the original algorithm, including structural changes to allow for seismic images as inputs, a change in the way edges are constructed for the graph, and a change in the weighting calculation for each edge. Each of these modifications increased the accuracy of the method when applied to seismic data.

Initial results from applying the modified algorithm to both synthetic and field seismic images are extremely encouraging. Segmentation of a synthetic image accurately located the boundaries of a complex salt body, although several different segments were required. Segmentation of both 2D and 3D field seismic data was even more successful. Compared to an existing implementation of the Normalized Cuts algorithm from Shi and Malik (2000), the new method performs extremely well – it required only half a minute to segment the synthetic data image (which is too large for the NCIS implementation to handle on a single processor), and only one second to segment the smaller field data example, over 150 times faster than NCIS. An additional advantage is that the newer algorithm is able to operate on the entire image, rather than only within a certain windowed radius of a previously interpreted boundary. This approach has many advantages, not least of which is the opportunity to identify segments other than only salt bodies. Instead of a binary salt/no-salt determination, the ability to identify coherent sedimentary "segments" as well would be tremendously useful for constructing seismic velocity models.

While these results are promising, there are many potential improvements that remain to be explored. First, the fact that the salt body is in some cases divided into multiple segments is a situation that should be examined. One possibility is to modify the edge weight calculation to include the relative importance of the amplitude/intensity and distance factors; right now, both terms are weighted equally (see equation 6). Another option is to change either the shape or size of the stencil (Figure 5) used to build the graph. Yet another potential improvement is the incorporation of multiple seismic attributes – for example, dip and frequency attributes. This enhancement can be accomplished relatively simply by taking multiple volumes as inputs, and calculating edge weights using some weighted combination of the different attributes at each pixel.



(a)



(b)

Figure 10: Slices through a 3D image (a) and the resulting 3D segmentation (b) using the new PRC algorithm. [**ER**]

Interpreting subsurface salt bodies in large, 3D seismic images is an incredibly complex and tedious task. With further improvement, however, an accurate, *efficient* automatic segmentation scheme such as this one has the potential to be an extremely useful and powerful tool for processing and interpreting seismic images.

ACKNOWLEDGMENTS

I thank SMAART JV and Unocal (now Chevron) for providing the data used for examples in this paper, and Robert Clapp for many helpful suggestions.

REFERENCES

Felzenszwalb, P. F., 2010, Image segmentation. (`http://people.cs.uchicago.edu/ pff/segment/`).

Felzenszwalb, P. F. and D. P. Huttenlocher, 2004, Efficient graph-based image segmentation: International Journal of Computer Vision, **59**, 167–181.

Halpert, A., R. G. Clapp, and B. L. Biondi, 2009, Seismic image segmentation with multiple attributes: SEG Technical Program Expanded Abstracts, **28**, 3700–3704.

Lomask, J., 2007, Seismic volumetric flattening and segmentation: PhD thesis, Stanford University.

Lomask, J., R. G. Clapp, and B. Biondi, 2007, Application of image segmentation to tracking 3d salt boundaries: Geophysics, **72**, P47–P56.

Shi, J. and J. Malik, 2000, Normalized cuts and image segmentation: Institute of Electrical and Electronics Engineers Transactions on Pattern Analysis and Machine Intelligence, **22**, 838–905.

Taner, M. T., F. Koehler, and R. E. Sheriff, 1979, Complex seismic trace analysis: Geophysics, **44**, 1041–1063.

Zahn, C. T., 1971, Graph-theoretical methods for detecting and describing gestalt clusters: IEEE Transactions on Computers, **C-20**, 68–86.