Chapter 2

Interpreter-guided seismic image segmentation

Salt interpretation is a vital component of seismic imaging projects in many of the world's resource-rich basins. The sharp contrast between seismic velocities within salt structures and those in the surrounding sediments means that inaccurate interpretation of these salt-sediment boundaries can lead to severe degradation of images sub-salt; this is of particular concern since sub-salt reservoirs are often the targets for modern exploration. Unfortunately, salt interpretation is not only critical, but often time-consuming and human-intensive as well. For large 3D surveys, manual salt-picking can consume significant resources during model-building workflows that stretch for weeks or months. This can be exacerbated by iterative sediment- and salt-flooding techniques that require several rounds of salt interpretation (Mosher et al., 2007). The semi-automatic image segmentation method presented here aims to help alleviate this bottleneck, while maintaining the accuracy necessary for successful model building and imaging.

SEGMENTATION METHODS

While image segmentation is most often associated with fields such as medical imaging and photo processing, several efforts have been made to apply automatic segmentation concepts to seismic images. A variety of approaches has been tried, including pixel-by-pixel classifier methods using fuzzy math (Valet et al., 2001) or texture attributes (Berthelot et al., 2012). These methods can incorporate interpreter input by "training" the algorithm through the use of if-then guidelines or training images. Another category of methods that has proven popular for seismic images is known as 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 (for example, a partition may represent a salt boundary). An advantage of graph-based segmentation is that it provides a globally optimum solution to the segmentation problem. This compares favorably with automatic interpretation tools such as horizon trackers that tend to get "lost" if a boundary becomes chaotic or discontinuous. Recall the example from the previous chapter (Figure 1.2), in which one such horizon tracker fails to follow a prominent salt boundary, even when several seed points are provided.

The first graph partitioning seismic image segmentation algorithms were adapted from the eigenvector-based Normalized Cuts Image Segmentation (NCIS) method (Shi and Malik, 2000). This method is based on transforming an image into a normalized eigenvector, with values ranging from -1 to 1 across a boundary. One of the first applications was for atomic meshing of seismic images (Hale and Emanuel, 2002, 2003), followed by efforts to track salt boundaries (Lomask et al., 2007; Lomask, 2007; Halpert et al., 2009). The method was effective, but faced limitations - most notably computational. The NCIS algorithm requires the calculation of eigenvectors 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.

PRC SEGMENTATION

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.

The algorithm proposed by Felzenszwalb and Huttenlocher (2004) relies heavily on the concept of the "Minimum Spanning Tree" [see Zahn (1971)]. To understand the minimum spanning tree, a brief overview of graph-partitioning theory is helpful. Recall that a graph is composed of vertices (an image's pixels) connected by edges. The graph's edges may be weighted using a measure of similarity or dissimilarity between vertex pairs; for example, a simple weighting could simply difference intensity values at the two endpoints of an edge. A *connected* graph (example (a) in Figure 2.1) is defined as one in which all such edges are assigned a weight value. A spanning tree (Figure 2.1(b)) is a connected graph which connects all its vertices without closing a circuit. Finally, the minimum spanning tree (MST) of a graph is the spanning tree with the minimum sum of edge weights (Figure 2.1(c)). 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, the heterogeneous nature of the intensity values within a salt body). However, the MST concept allows Felzenszwalb and Huttenlocher (2004) to develop what they term a "pairwise region comparison" predicate, based on three definitions described below and illustrated in Figure 2.2. The *internal difference* of a region (C) in the graph to be the largest edge weight of the MST of that region:

$$\mathrm{Int}(C) = \max_{e \in \mathrm{MST}} w(e), \tag{2.1}$$

where e is a graph edge and $w(e)$ is the edge's weight, defined according to some simple algorithm. In Figure 2.2(a), there are two groups of pixels that the algorithm must either separate or merge. The internal difference is computed for each group. When comparing two regions (such as C_1 and C_2), the *minimum* internal difference for the two regions is

$$\mathrm{M}_{\mathrm{Int}}(C_1, C_2) = \min(\mathrm{Int}(C_1) + \tau(C_1), \mathrm{Int}(C_2) + \tau(C_2)), \tag{2.2}$$

where τ is a positive 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. In Figure 2.2(b), the minimum internal difference is determined by comparing the internal difference values for each group of pixels. Finally, the *difference* between the two regions, shown in Figure 2.2(c), is the smallest edge weight that connects them:

$$\mathrm{Dif}(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2} w((v_i, v_j)), \tag{2.3}$$

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, the algorithm simply compares the values of $\mathrm{Dif}(C_1, C_2)$ and $\mathrm{M}_{\mathrm{Int}}(C_1, C_2)$. If $\mathrm{Dif}(C_1, C_2)$ is greater, the "pairwise comparison predicate" is determined to be true, and the two regions are separated. If the predicate fails, the two regions are merged.

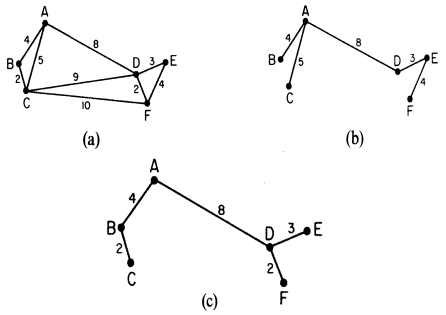
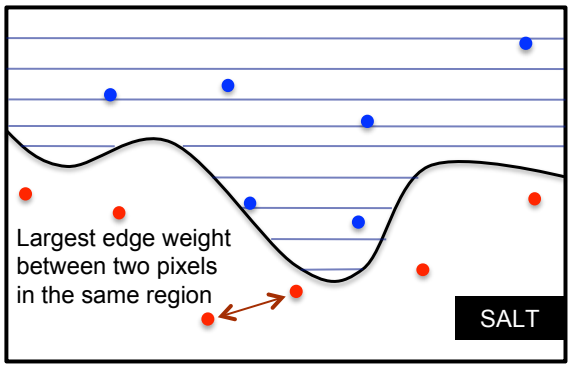


Figure 2.1: Modified from Zahn (1971). A graph with weighted edges (a); a spanning tree of that graph (b); and the minimum spanning tree of the graph (c). [**NR**] chap2/. MST

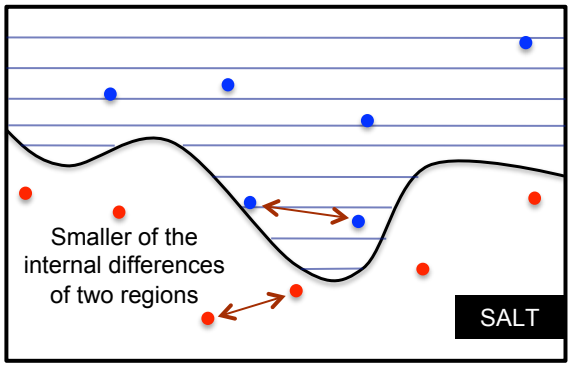
In Figure 2.2, the three definitions above are used to partition two groups of pixels separated by a putative salt boundary. While this is a relatively simple procedure, it is designed to allow highly heterogeneous regions to be segmented as a single component of an image – an important capability when handling noisy images. 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.

ADAPTATION FOR SEISMIC IMAGES

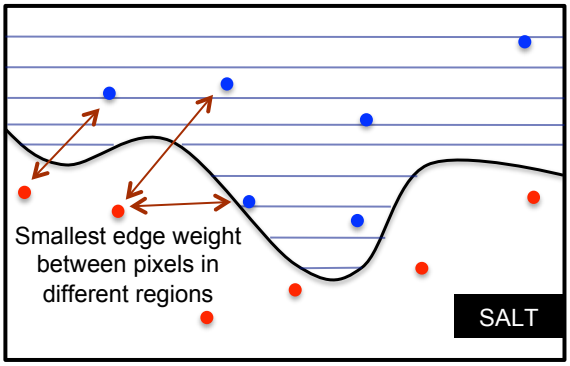
Seismic images are distinct in many ways from conventional photographs and medical images for which this and most image segmentation algorithms are designed. The consequences of these differences can be seen in Figure 2.3(b), the result of using the un-altered PRC algorithm to segment a 2D image from the Gulf of Mexico (Figure 2.3(a)). In this and all subsequent depictions of segmentation results, the interpreted segments are assigned a random color and overlaid on the image for reference.



(a)

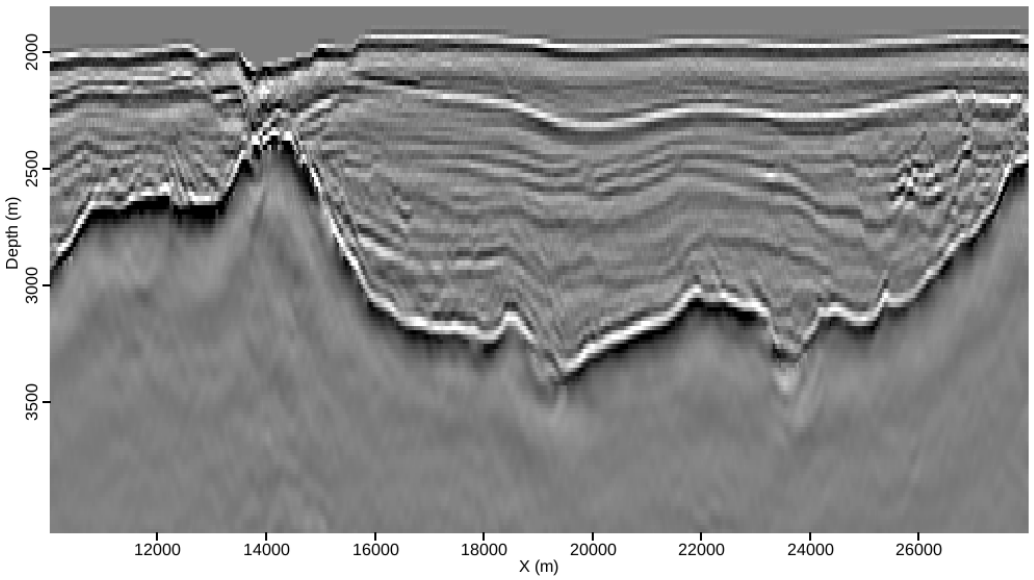


(b)

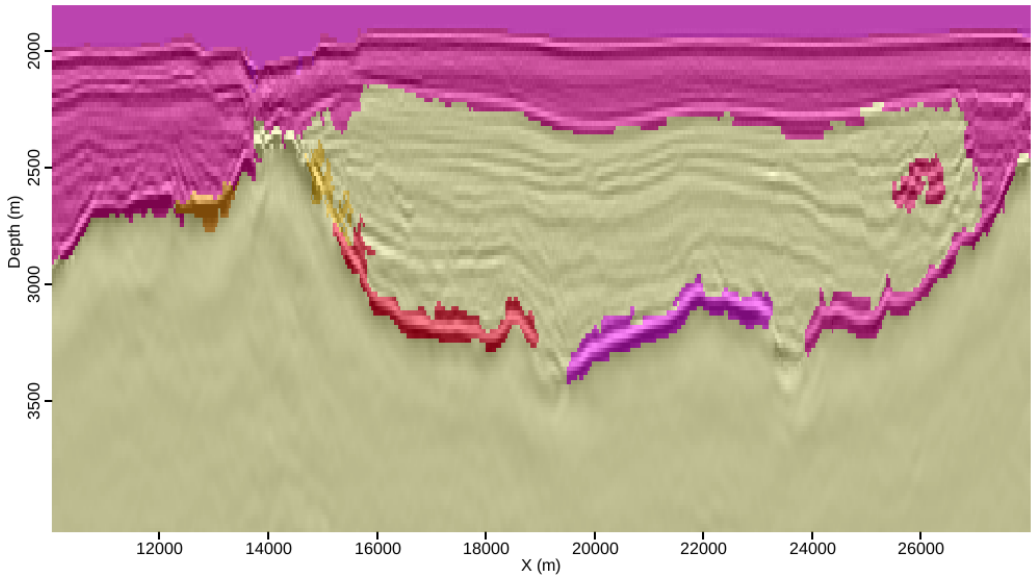


(c)

Figure 2.2: Illustrations of three crucial definitions for PRC segmentation, on two groups of pixels that may be separated by a boundary. (a) The *Internal difference* of a group of pixels is the largest edge weight between two pixels in that group; (b) the *Minimum internal difference* is the smaller of the internal differences from each group; (c) the *Difference* is the smallest edge weight between two pixels in different groups. [**NR**] chap2/. Int,MID,Dif



(a)



(b)

Figure 2.3: (a) A 2D field seismic image, and (b) its corresponding segmentation using the original algorithm from Felzenszwalb and Huttenlocher (2004). Prior to modification, the algorithm performs poorly because of the unique characteristics of seismic data. [**NR**] chap2/. uno-img,uno-origseg2

Transformation of input data

An initial hurdle is that 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 2.3(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:

$$A(z)e^{i\phi(z)}, \tag{2.4}$$

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 amplitude is the only information present, the problem described above may be avoided. Figure 2.4 displays such an envelope calculation for the image in Figure 2.3(a).

Edge-preserving smoothing

In many disciplines, pre-processing images by smoothing is common prior to automatic image segmentation; however, traditional smoothing blurs boundaries and removes high spatial-wavelength features, which is counterproductive for seismic imaging and interpretation. For example, Figure 2.5(b) is the result of performing traditional box-filter smoothing on the 3D image in Figure 2.5(a). However, an edge-preserving smoothing technique based on a method called "directional maximum homogeneity" (Zahedi and Thomas, 1993) can remove unwanted incoherent noise from

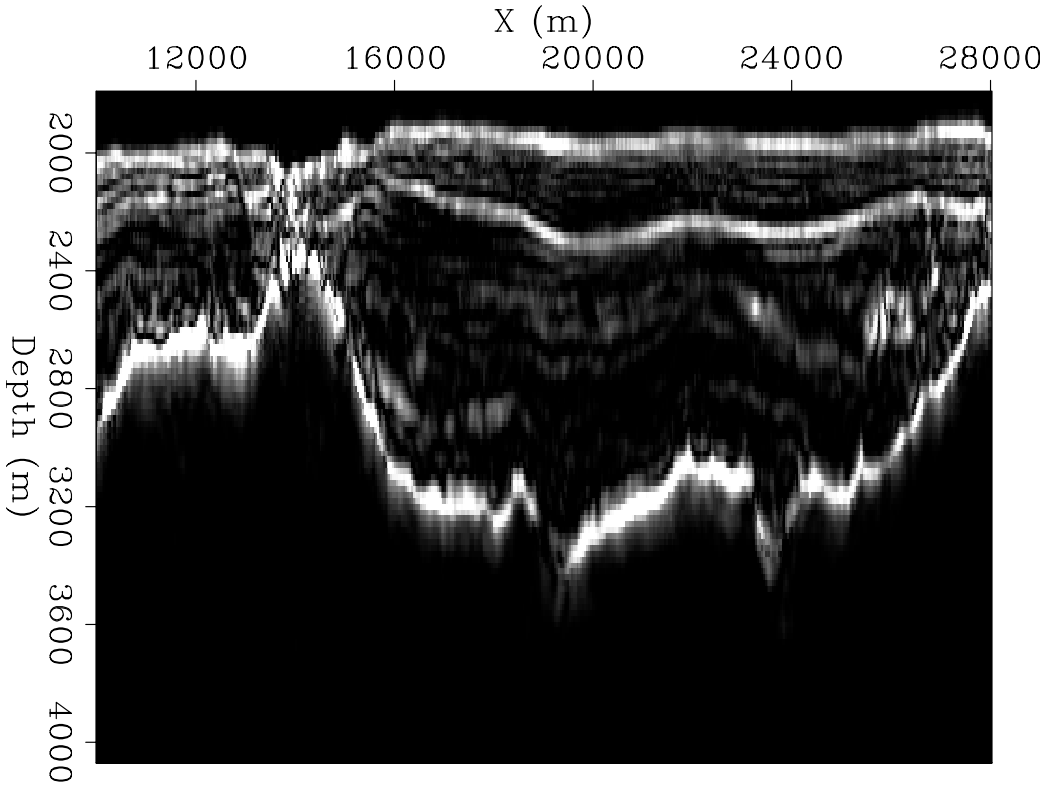
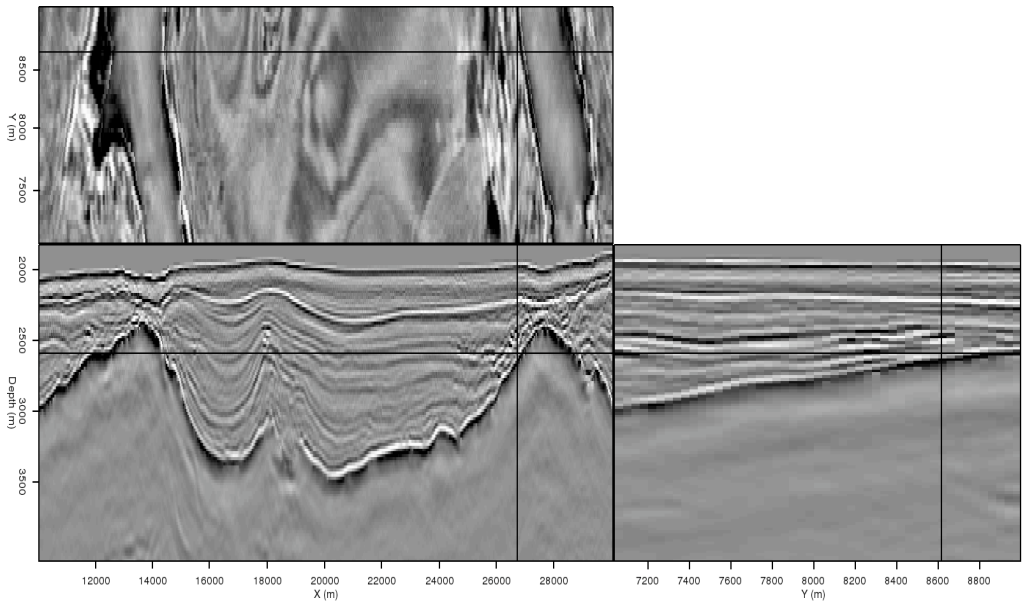


Figure 2.4: Amplitude of the envelope calculation for the raw image in Figure 2.3(a). This becomes the input for the PRC segmentation algorithm. [**ER**] chap2/. uno-env

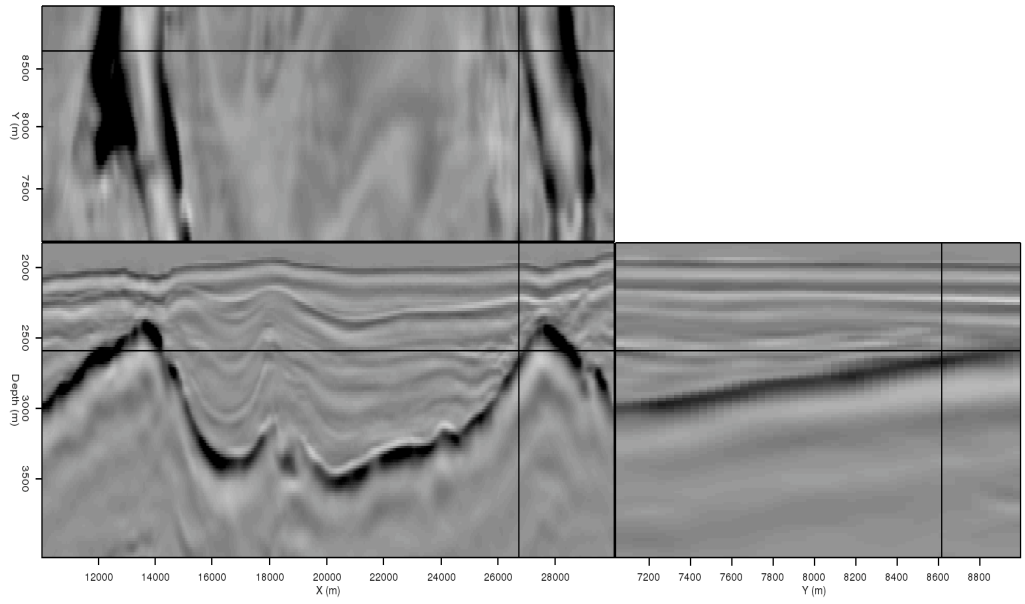
an image, while maintaining the sharp boundaries desirable for accurate segmentations. The 3D smoothing method constructs nine 1D "bar masks" extending in all directions from a central pixel, and calculates the variance of the amplitude values in each bar. By assigning the median value of the most homogeneous bar mask (smallest variance value) to the central pixel, the algorithm avoids smoothing across coherent boundaries. An indication of the superiority of edge-preserving smoothing can be seen in Figure 2.6. Differencing the image smoothed using a box filter from the original image yields a result that looks very much like the original image, indicating that a great deal of information (signal) has been removed. However, the difference result using the edge-preserving smoothing technique removes mostly "speckle" noise and very little of the signal.

Further smoothing can be accomplished by introducing a hybrid approach that combines the characteristics of box smoothing in areas without edges, and takes advantage of the edge-preserving features of maximum-homogeneity (MH) filtering when edges are present. For the MH algorithm described above, we already calculate variances for each of the bar masks passing through a given pixel. Comparing the largest and smallest of these calculated variances indicates the likelihood that an edge is present. If the ratio between the smallest and largest variances is large (close to 1), the pixel is in a relatively "isotropic" area, and an edge is unlikely to be present. Conversely, a smaller ratio implies that an edge is present in at least one of the bar mask orientations. If α is a user-determined threshold value, then if $\frac{\min(\sigma)}{\max(\sigma)} > \alpha$, traditional smoothing can safely be used in lieu of MH filtering for that particular location.

Figure 2.7 demonstrates this strategy on the 3D field data example. As the threshold value α decreases, the algorithm is more biased toward traditional smoothing; consequently, the amount of "speckle" noise decreases. Even at the smallest value of α (Figure 2.7(d)), the result is still superior to that of the traditional smoothing approach (Figure 2.5(b)) – even though the length of the bar masks and the dimensions of the box filter are equivalent. Smoothing 3D images in this manner allows the PRC segmentation algorithm to behave more robustly.

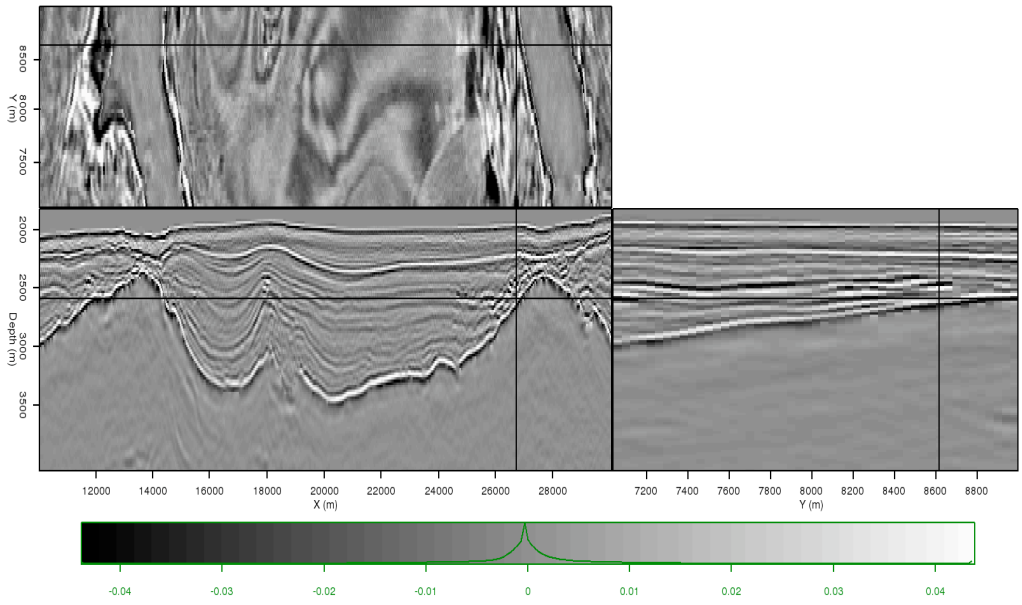


(a)

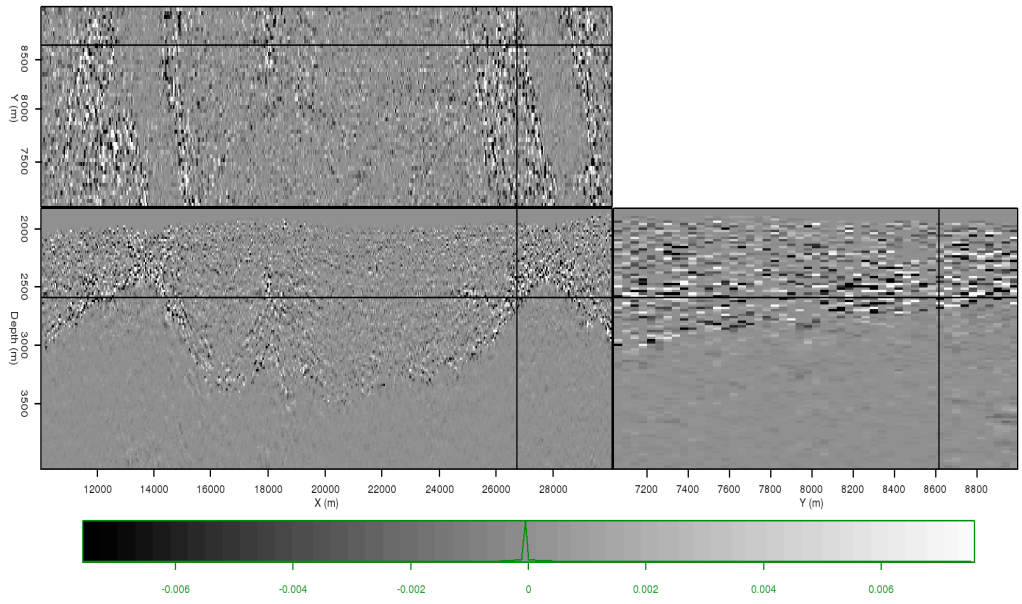


(b)

Figure 2.5: (a) A 3D image from the Gulf of Mexico; (b) The result of smoothing the image in panel (a) with a standard box-filter. [**ER**] chap2/. img-orig,img-smth



(a)



(b)

Figure 2.6: Results of differencing the original image in Figure 2.5(a) with a smoothed version obtained using (a) standard box-filtering, and (b) directional maximum homogeneity filtering. Both examples use the same size operator (box side length or bar mask length, respectively). [**ER**] chap2/. diff-smth,diff-mhn

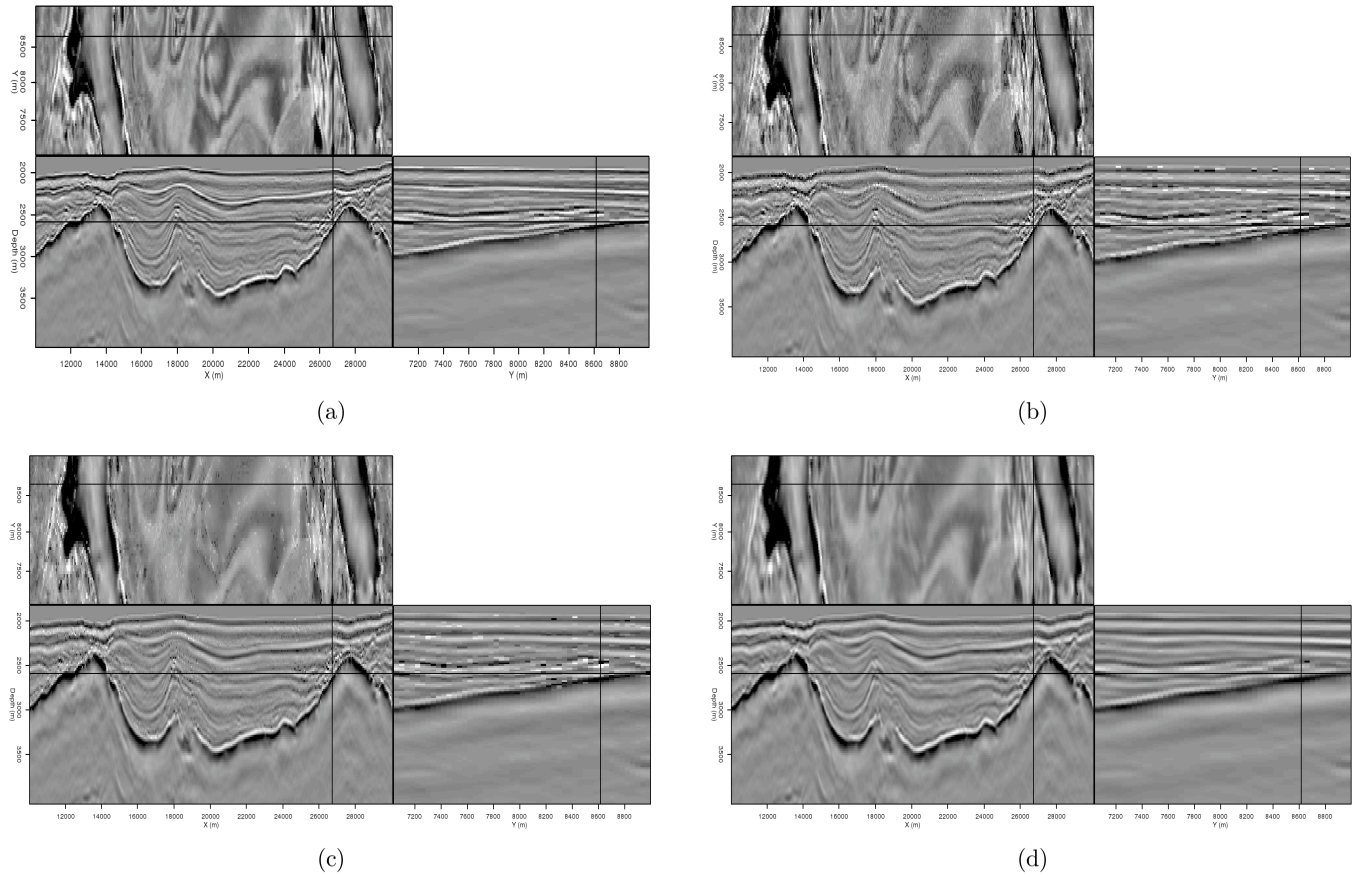


Figure 2.7: Results of applying the hybrid-MH filter on the image in Figure 2.5(a), with α set at (a) 0.5, (b) 0.2, and (c) 0.1 and (d) 0.01. While the reflector amplitudes are affected, a great deal of speckle noise is removed at low α values. [**ER**] chap2/. img-hyb-50,img-hyb-20,img-hyb-10,img-hyb-01

Edge construction

A distinguishing feature of seismic images is that regions such as salt bodies are most easily delineated by their boundaries, rather than, for example, color attributes used to segment photographs. Therefore, modifications to the algorithm's procedure for both constructing the graph and weighting its edges are required to obtain acceptable segmentation results for seismic images. The original implementation of the PRC algorithm 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 2.8 illustrates this process – if the "active" pixel is the one in red, edges are 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 larger stencil, such as the one on the right in Figure 2.8, has been implemented. The length of the stencil's arms is a user-defined parameter which may be adjusted based on data quality; larger stencils should be used for noisier data, but the trade-off is increased computational complexity. Increasing the size of the stencil allows for many more comparisons per pixel, and a far greater amount of information goes into the segmentation algorithm. 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 increased number of edges per node, this algorithm is still far less computationally intensive than the NCIS algorithm from Shi and Malik (2000).

Edge weighting

Finally, the edges constructed using the modified stencil in Figure 2.8 must be weighted in a manner that treats a boundary *between* two vertices as more convincing evidence for the existence of two regions than simply a difference in intensity at the

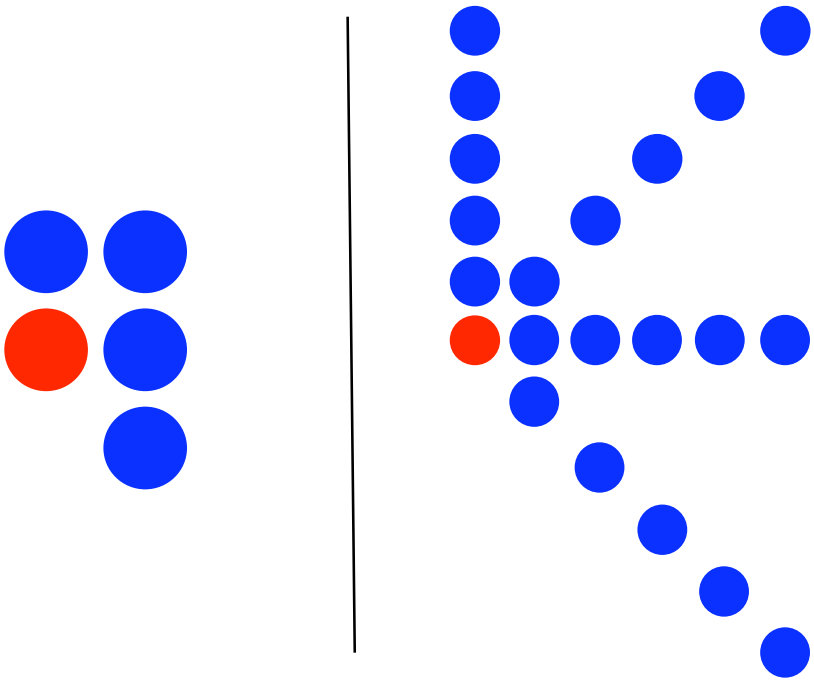


Figure 2.8: Stencils used for comparing pixel values and assigning edge weights for the graph corresponding to a 2D image. At left, the five-point stencil (8 edges per pixel) used in the original implementation from Felzenszwalb and Huttenlocher (2004); at right, a modified 21-point stencil (40 edges per pixel) used for the seismic images. For 3D images, additional stencil arms extend into the third dimension; the length of the stencil arms is a user-defined parameter. [**NR**] chap2/. stencils

two pixels themselves. When determining the weight for an edge with an endpoint along one arm of the stencil in Figure 2.8, I use the largest intensity value of any pixel between the two endpoints. For example, a high intensity value along one arm of the stencil would suggest that that particular arm intersects a boundary. Figure 2.9 illustrates the logic behind this process.

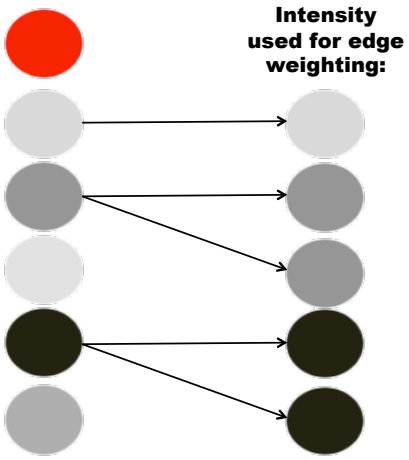


Figure 2.9: Diagram illustrating the logic behind deciding which pixel intensity value to use when calculating edge weights. Pixel intensities along one "arm" of the stencil in Figure 2.8 are shown on the left; darker colors represent higher intensities. The right column indicates which intensity value will be used when calculating the edge weight between the "active" (red) pixel and the adjacent pixel. [**NR**] chap2/. weights

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 2.9. 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.

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
```

Once we have selected the intensity value to use for determining the edge weight, the weight value is calculated using an exponential function:

$$w_{ij} = \exp((\max I(\mathbf{p}_{ij}))^2) \exp(d_{ij}), \tag{2.5}$$

where $\mathbf{p}_{ij}$ is the vector of all pixels between i and j and d_{ij} is simply the Euclidean distance (in samples) between the two pixels. Recall that the $\max I(\mathbf{p}_{ij}))$ term is determined according to algorithm 1. The distance-weighting d term accounts for the fact that the edges in the graph can now be much longer than with the adjacent-pixels-only approach taken in the original implementation.

Once each of the edges is assigned a weight, the segmentation of the image can proceed as described in Felzenszwalb and Huttenlocher (2004). Algorithm 2 summarizes this process, which begins with each pixel as its own image segment. Then, individual pixels, and eventually, groups of pixels, are merged according to thresholding criteria. Segments can also be merged in post-processing if they are smaller than a "minimum segment size" parameter specified by the user. Figure 2.10 is the much-improved result when the example image in Figure 2.3(a) is segmented by the modified PRC algorithm.

Algorithm 2 PRC segmentation procedure

```
for each pixel ipix in image do
    build edges to neighboring pixels;
    weight the edges according to the procedure in algorithm 1;
end for
sort the edges in increasing order;
create initial segmentation S_0
for each edge iedge in the sorted list do
    calculate Dif and MInt values for pixels/regions connected by iedge
    if Dif > MInt then
        partition the image at iedge
    else
        merge the two pixels/regions
    end if
end for
```

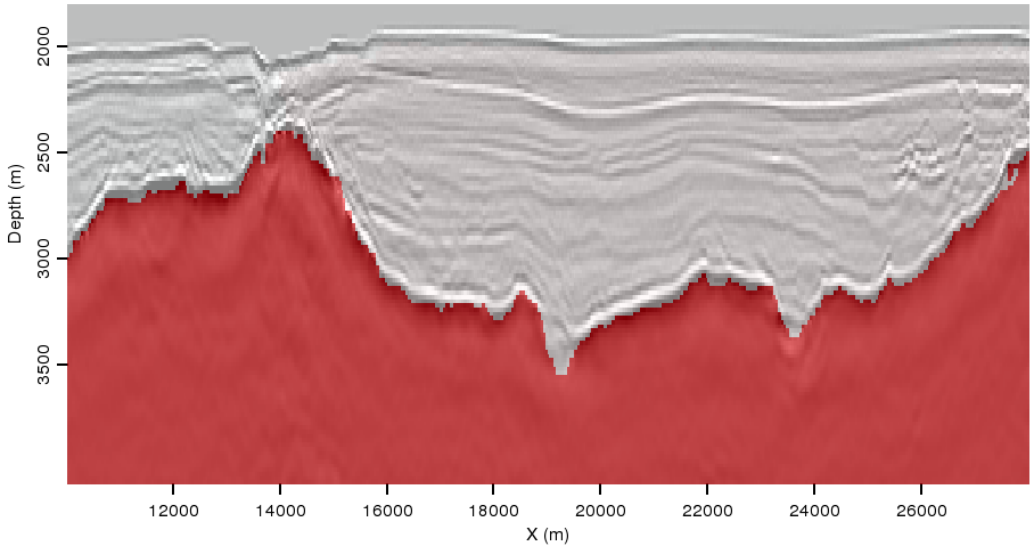


Figure 2.10: Final segmentation result of the image in Figure 2.3(a), after modifications to the algorithm. [**ER**] chap2/. uno-segmerge

PARAMETERIZATION AND UNCERTAINTY

A key advantage of the PRC segmentation scheme is ease of parameterization. In practice, only two user-controlled parameters must be set prior to segmentation: the length of the modified stencil arms described in the previous section, and the minimum size (in pixels) of the automatically interpreted image segments. The latter parameter is straightforward for salt segmentation, especially if a prior model provides an approximate size for the salt body. By setting this parameter to a value of the same order but smaller than the expected size of the salt body, an interpreter can maximize the likelihood of an accurate segmentation. It is advisable to err on the side of smaller minimum segment sizes; doing so may lead to a salt body being broken up into multiple segments, but this is easily remedied and creates an opportunity for interpreter guidance in difficult areas, as described in the next section.

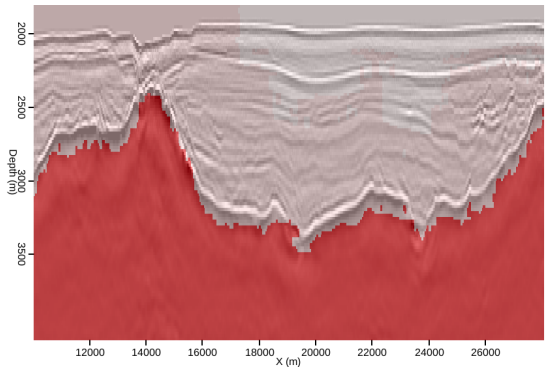
The effects of changing the stencil length parameter are more subtle, but may provide an opportunity for uncertainty analysis. The lack of a direct measure of degree of certainty or uncertainty for automatic segmentations is the primary drawback of the PRC method, compared, for example, to the NCIS method, in which the sharpness of the transition from negative to positive eigenvector values across a boundary acts as a proxy for certainty of the boundary location. While the PRC method does not provide a similar measure, the extreme efficiency of the algorithm opens the possibility of performing multiple segmentations of the same image, but with different parameters. By observing whether and where a salt boundary changes as a result of modifying parameters, we can hope to gain an understanding of where the interpreted boundary exhibits the most or least uncertainty. Because the stencil length parameter in effect controls the amount of information used to construct and weight graph edges for a given location, it is an ideal candidate for such a scheme.

Figure 2.11 shows segmentation results for the image in Figure 2.3(a), using three different values for the stencil arm length parameter. Physically, the length of the stencil arms can be related to the wavelength of the salt boundary reflector. In this case, the width or wavelength of the salt boundary throughout the image is

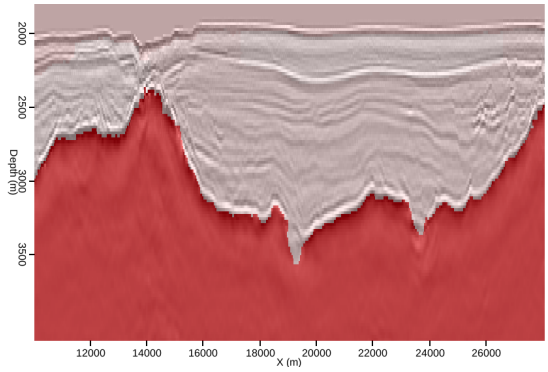
approximately five pixels, and employing a stencil of this length appears to produce the most accurate result (panel b). A larger stencil (panel a), in which the arms extend for approximately twice the wavelength of the boundary reflector, provides the least accurate result. This makes intuitive sense, as even pixels far from the boundary may be affected by the boundary's "halo". However, a smaller (three-pixel) stencil with arms extending approximately half of the reflector's wavelength (panel c) performs almost as well as the ideal five-pixel stencil, with the exception of a clear error near $x = 23000m$. At this location, the additional information taken in by the longer stencil, at increased computational cost, is a worthwhile tradeoff. Interestingly, the boundary at this locations is very faint and even discontinuous compared to the rest of the image; this can be seen not only on the underlying image in Figure 2.11, but especially on the corresponding amplitude of the envelope image in Figure 2.4. In this example, varying the stencil length parameter has quickly and clearly identified a zone of high uncertainty along the boundary.

INTERPRETER GUIDANCE

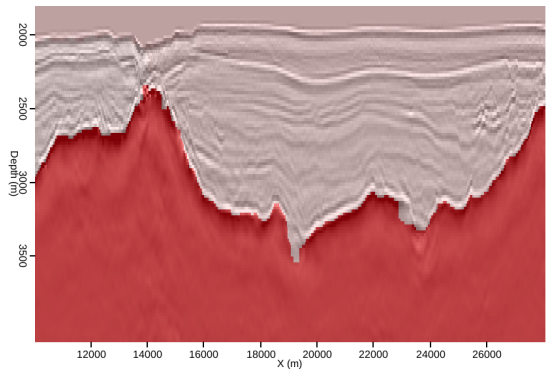
Unfortunately, a fully-automatic method will often be insufficient for obtaining an acceptable salt interpretation. In such cases, results of the automatic process can be evaluated and improved via input from a human interpreter. Interpreter input is desirable for a variety of reasons. First, in many circumstances, interpreters can spend years learning the geological features of a given basin or area of the world; this knowledge is extremely valuable for model-building purposes, and near impossible to "program" into a fully-automated procedure. A second reason human input is valuable stems from the pattern recognition advantages of humans relative to computers, especially in two dimensions. Even novice interpreters would likely have little trouble immediately recognizing the salt body on the 2D section in Figure 2.12, even though, as we will see shortly, automatic interpretation algorithms may struggle. However, in three (or more) dimensions, taking advantage of computational resources could allow interpreters to overcome visualization challenges while streamlining the interpretation process. Ideally, a semi-automatic interpretation workflow would allow interpreters



(a)



(b)



(c)

Figure 2.11: Automatic segmentation results for the image in Figure 2.3(a), using three different lengths for the stencil arms depicted in Figure 2.8: (a) 10 pixels; (b) 5 pixels; (c) 3 pixels. [**ER**] chap2/. stencil10,stencil5,stencil3

to provide limited manual interpretations on 2D sections, and use that information to guide the automatic 3D seismic image segmentation. This is the goal of the method presented here.

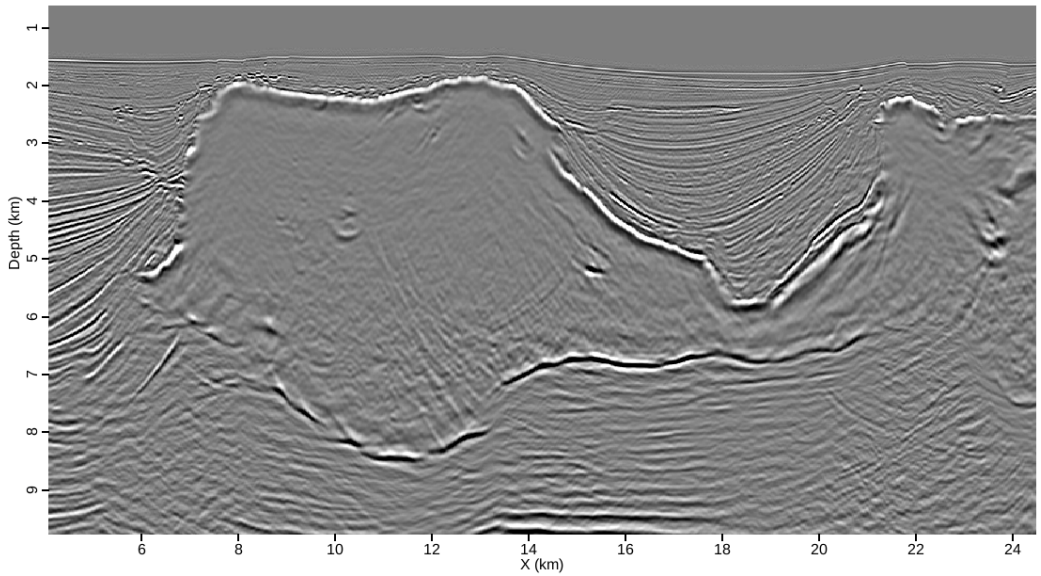
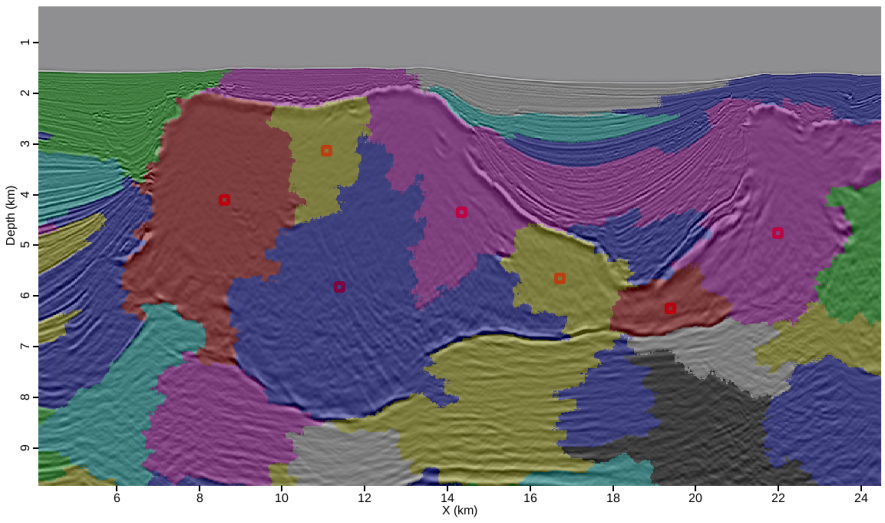


Figure 2.12: A 2D section from a Gulf of Mexico dataset. The prominent salt body provides an ideal test case for segmentation algorithms. [**ER**] chap2/. oct-2d

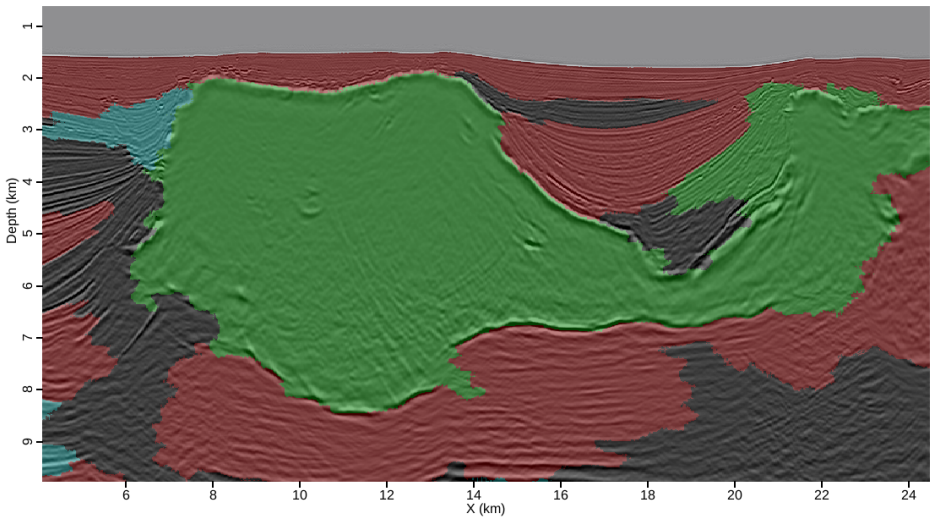
An initial opportunity to incorporate interpreter guidance is apparent from the segmentation result in Figure 2.13(a), obtained using another 2D example from the Gulf of Mexico (Figure 2.12). In this result, the algorithm has identified several distinct segments within the salt body. It would be much easier to evaluate the segmentation results if the entire salt structure were instead a single segment. This is easily achieved if an interpreter quickly identifies which segments belong inside the salt; a single click can accomplish this task, as shown by the red dots in Figure 2.13(a). Merging these segments provides a much more interpretable result, seen in Figure 2.13(b). An additional benefit of this capability involves the model-building aspect of segmentation. Assigning a user-defined value to the new, merged segment allows for a simple means of setting velocities for specific regions, such as a salt body. Furthermore, in some cases there may be a question of whether or not a certain segment actually belongs in a salt body; by choosing to include or exclude the segment, an interpreter can quickly generate multiple possible models. This is ideal for scenario testing, an issue I will revisit in Chapter 4.

Acquisition, model-building, and imaging challenges all contribute to situations in which salt boundaries appear faint, discontinuous, or not to be present at all. In Figure 2.12, for example, there are locations along both the top and base of the salt body where the boundary is poorly imaged. This can result in "leakage" of the automatically interpreted salt segments seen in Figure 2.13(b). In these cases, valuable interpreter insight should be incorporated into the procedure. Figure 2.14 shows manual salt boundary interpretations in areas where leakage is apparent in Figure 2.13(b). The most efficient way to include this information in the PRC algorithm is to modify the input image by increasing intensity values at the manual pick locations. Instead of assigning arbitrarily large values, however, we define a new amplitude value (A) for a manually-picked pixel at position (x,y,z) in terms of the highest-amplitude pixel in a neighborhood surrounding it and a scaling factor α:

$$A_{xyz} = \alpha \max_{|x-i|\leq 5, |y-j|\leq 5, |z-k|\leq 5} A_{ijk}. \tag{2.6}$$



(a)



(b)

Figure 2.13: (a) A segmentation of the image in Figure 2.12 according to the algorithm presented in section 2. The salt body is divided into several segments, so an interpreter can select which segments (denoted by the red dot) belong with the salt body. (b) Result after merging the segments marked in panel (a). It is now much easier to properly evaluate the segmentation results. [**ER**] chap2/. o2d-picks,2d-orig

This ensures that the picked boundary will not appear radically different from its surroundings, which could present challenges for the automatic segmentation algorithm. Now, segmenting the new input image with parameters identical to the original segmentation yields the result seen in Figure 2.15. The segments conform to the manual picks seen in Figure 2.13(b), while the rest of the image is segmented as accurately as the original result in Figure 2.13(b).

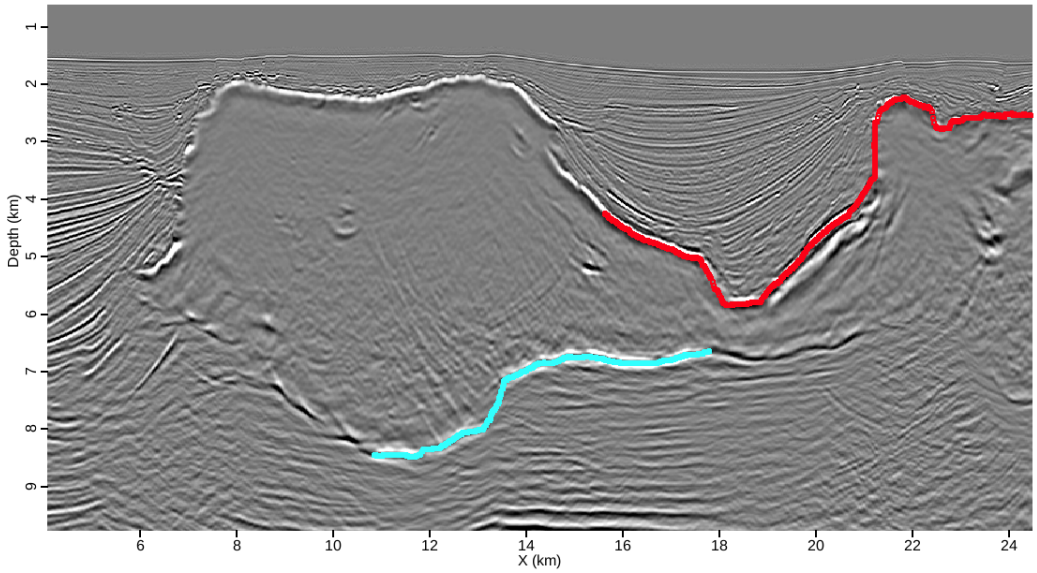


Figure 2.14: Manual salt picks supplied to guide the automatic segmentation. These picks are designed to correct the leakages seen on the segmentation result in Figure 2.13(b). [**ER**] chap2/. 2dtopbase

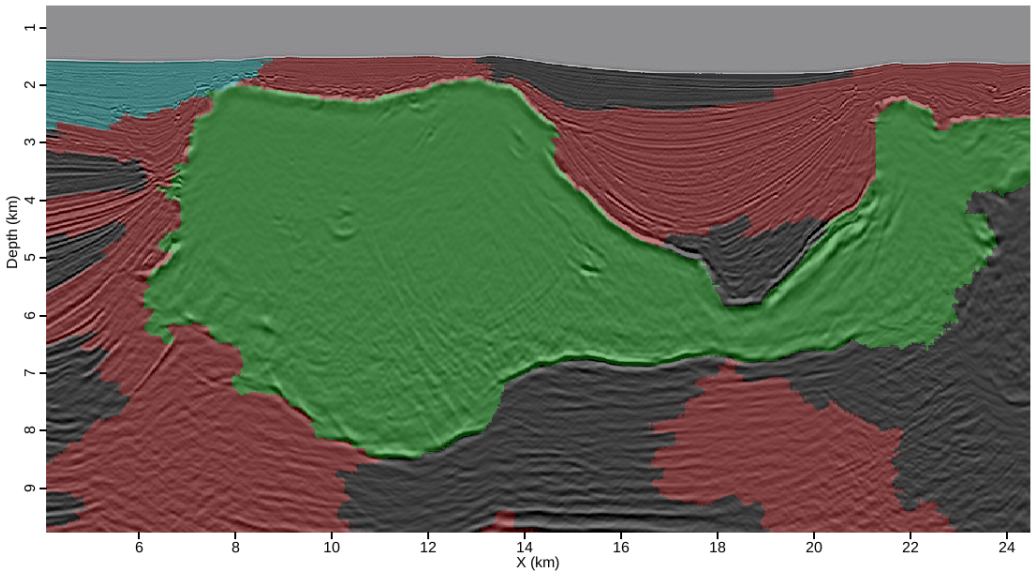


Figure 2.15: Segmentation of Figure 2.12 after incorporation of interpreter guidance. [**ER**] chap2/. seg-final

Additional modifications are required for 3D images. Because segments are much larger in 3D, amplitude changes on a single 2D section are not significant enough to alter 3D segmentation results. Instead, we must "project" an interpreter's manual picks on an inline section, into the third (crossline) dimension. To do this, we make the assumption that a salt boundary will not fluctuate by more than two pixels per slice in the crossline direction, and construct a square pyramid in the crossline direction like the one depicted in Figure 2.16. The pyramid has sides of length $2h$, where h is the number of crossline samples between the base of the pyramid and its apex, which is the manually interpreted point. Now, for any pixel Q that falls within a pyramid with an apex at point P, the new amplitude value at point Q is

$$A^Q_{\text{new}} = A^Q_{\text{orig}} + \frac{A_0}{||PQ||}, \tag{2.7}$$

where A_0 is the amplitude value at point P as determined by equation 2.6, and $||PQ||$ is the distance between the two points. The expression is additive to ensure that any hint of the boundary already present will not be overwhelmed by the interpretation on a nearby slice. In addition, the influence of the manually-picked points decays with distance from those locations, to allow for natural variations in geology.

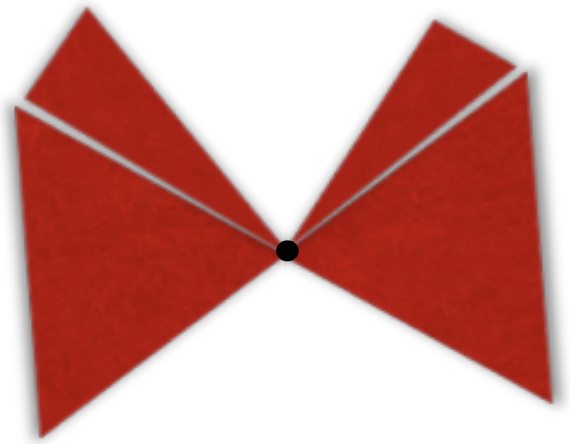


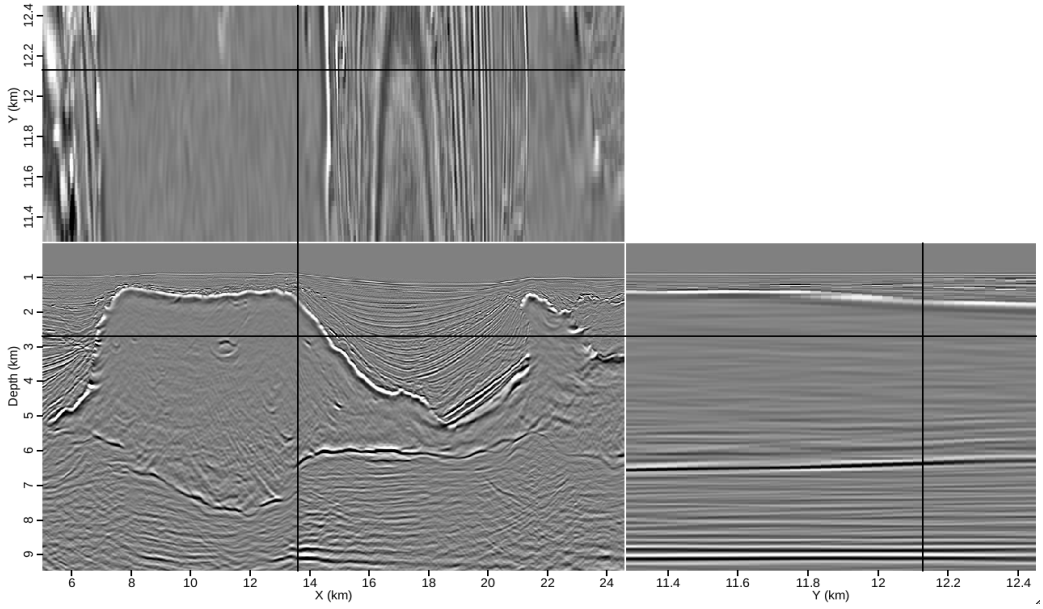
Figure 2.16: Depiction of the pyramid used to "project" an interpreter's picks from a single 2D slice into the third dimension. The influence of the pick (at the apex of the pyramid) decays with distance. [**NR**] chap2/. pyramid

3D field data example

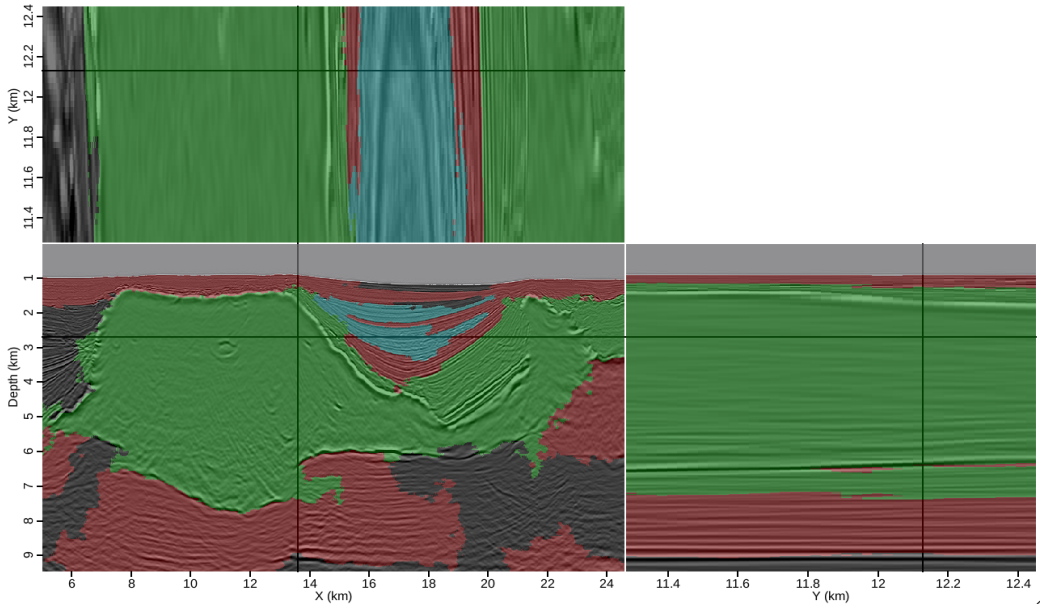
Figure 2.17(a) shows slices through a 3D image cube from a Gulf of Mexico dataset. From the initial segmentation result (Figure 2.17(b)), it is clear that the salt boundary discontinuities present challenges for the automatic segmentation algorithm. To correct the apparent leakages, manual interpretations are supplied for selected locations at two crossline values (Figure 2.18). The effects of these manual picks on the input amplitude data are shown in Figures 2.19(a) and 2.19(b). Not only are higher amplitudes obvious at the pick locations themselves, but the procedure described in the previous section has clearly influenced the intensity values at neighboring crossline values. Now, the updated segmentation result (Figure 2.20) is improved on both the inline and crossline sections. To emphasize the improvement, we should examine a location farther away from any manual picks. Figure 2.22(a) is the initial segmentation result for such a location, and leakages similar to those in Figure 2.17(b) are apparent. The effects of the 3D interpreter guidance scheme can be seen in Figure 2.21 – amplitudes have been boosted along the salt boundary in panel (b), even though the manual picks were supplied farther away. The improved segmentation result in Figure 2.22(b) demonstrates that even far from the actual picks, the automatic segmentation process is significantly more accurate when incorporating interpreter guidance. It is important to note, however, that the new segmentations are still not perfect; in particular, a small but noticeable inclusion within the salt body (near $X = 11km, Y = 11.5km$) is undetected. This is an artifact of the user-controlled minimum segment size parameter. In a model-building scenario like the one explored in Chapter 4, inclusions like this one can be isolated and accurately segmented.

COMPARISON WITH PREVIOUS METHOD

As expected, the algorithm is extremely efficient. This 3D example had over 30 million pixels, and over 700 million graph edges were constructed. In terms of cost functions, this translates to a computational requirement on the order of 5.2×10^8 operations for the PRC algorithm, compared to 9×10^{14} operations for the Normalized Cuts



(a)



(b)

Figure 2.17: (a) Slices through a 3D image cube from the Gulf of Mexico; (b) Segmentation result prior to interpreter guidance. [**ER**] chap2/. oct-3d,3d-origseg

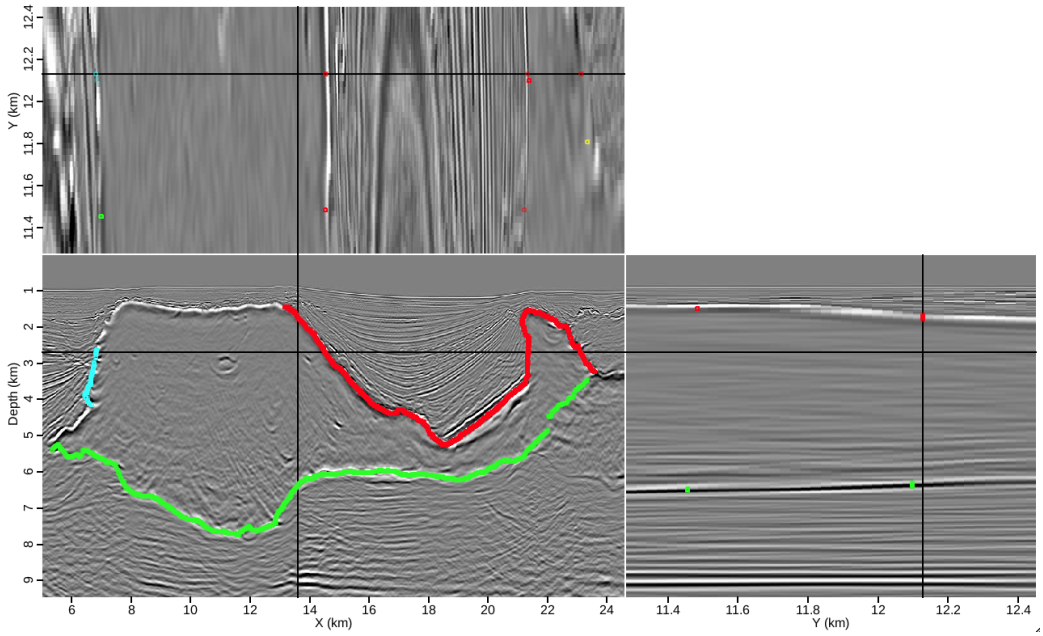
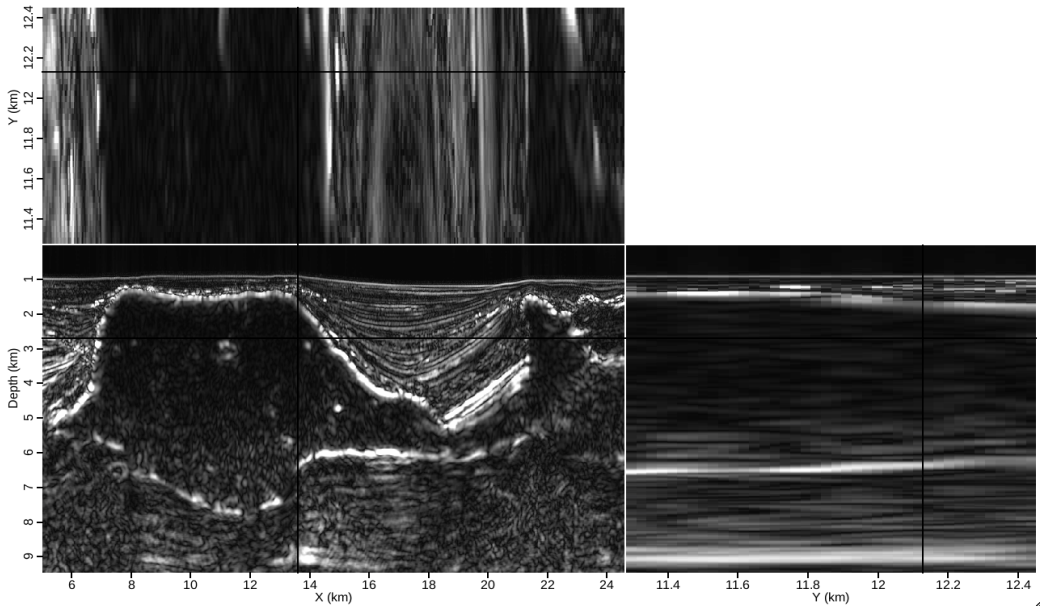
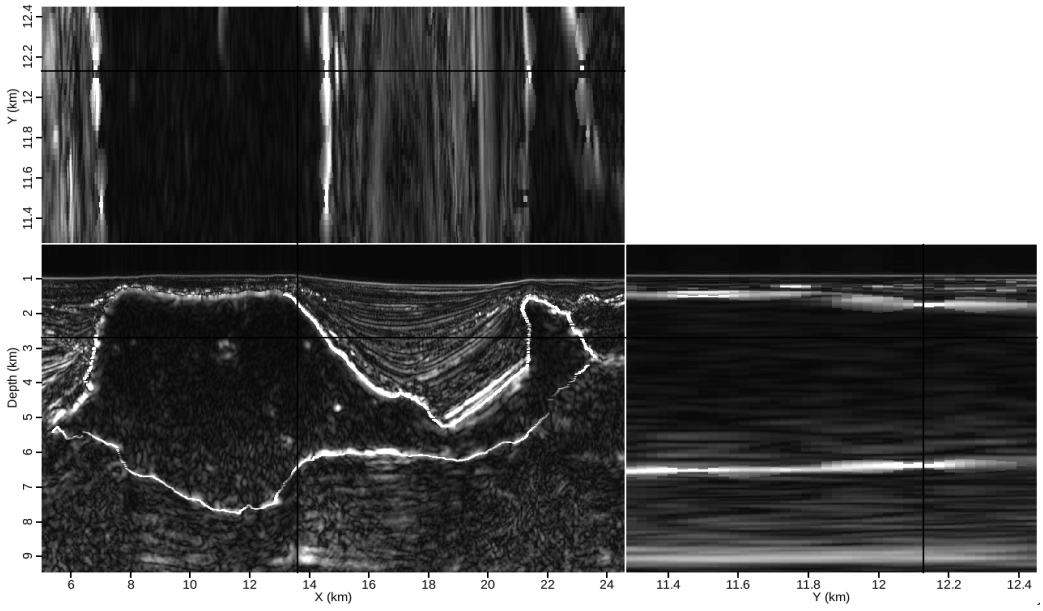


Figure 2.18: Manually-interpreted salt picks used to guide the automatic 3D segmentation. [**ER**] chap2/. oct-3d-picks



(a)



(b)

Figure 2.19: Amplitude of the envelope volumes (a) before and (b) after modification according to the interpreter guidance scheme. [**ER**] chap2/. 3d-env-orig,3d-env-new

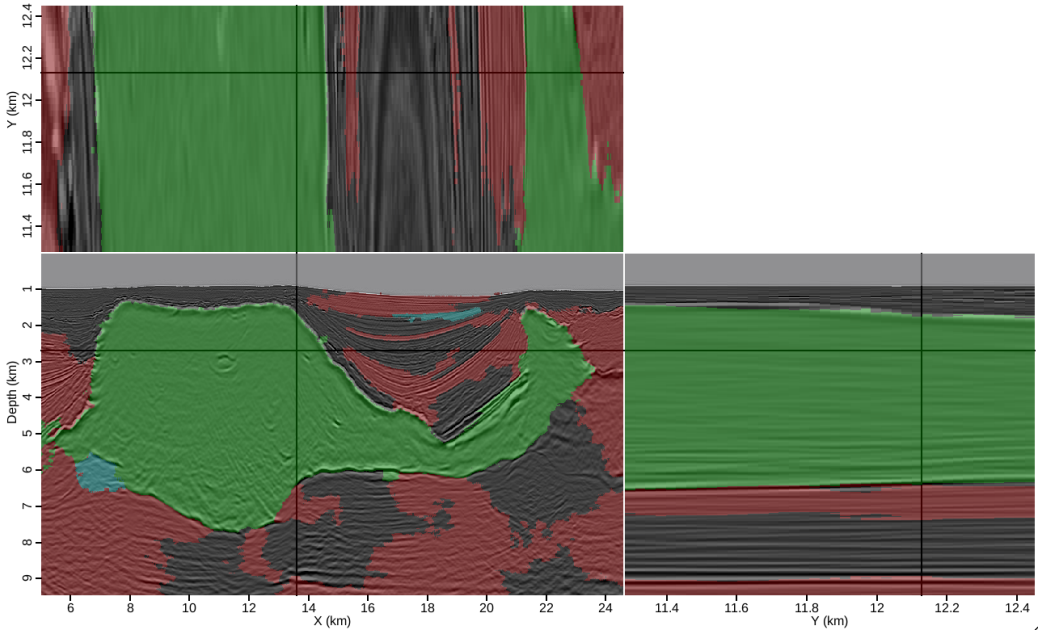
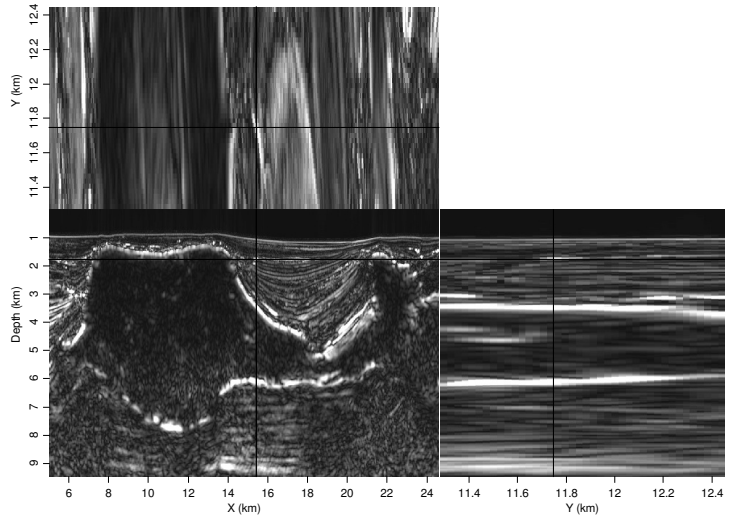
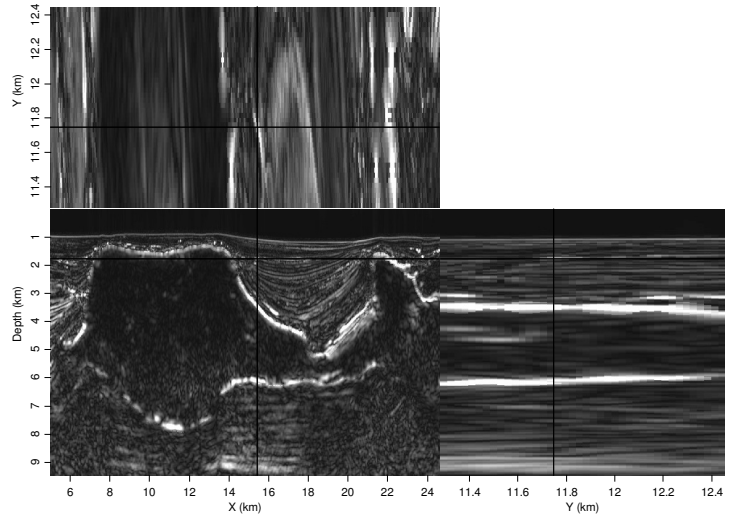


Figure 2.20: Segmentation result incorporating interpreter guidance. While not perfect, this result is much more accurate than the initial result in Figure 2.17(b). [**ER**] chap2/. 3d-newseg

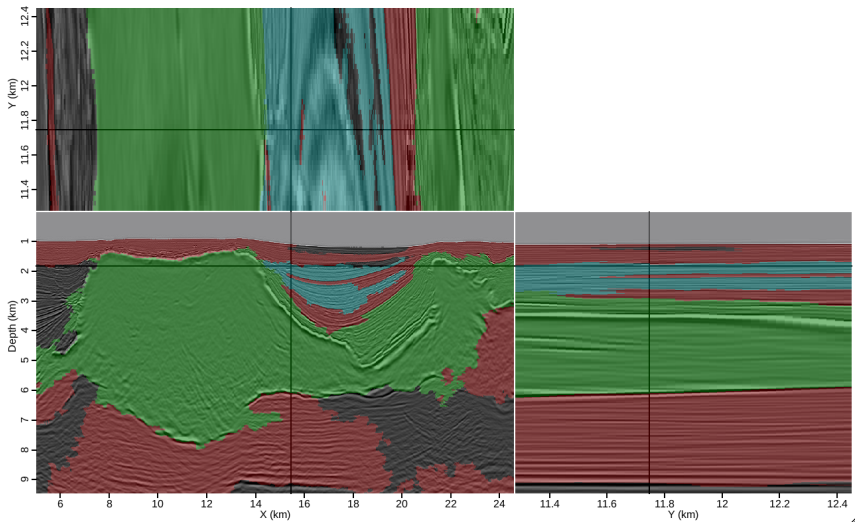


(a)

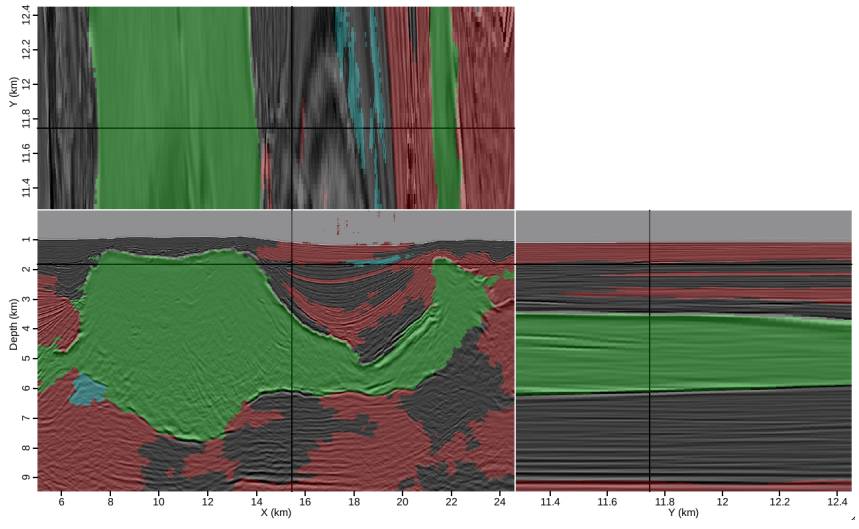


(b)

Figure 2.21: Amplitude volumes at a location far from any manual picks both (a) before and (b) after modification via 3D interpreter guidance. The effects of the interpreter guidance scheme can be seen along the salt body boundaries, even though the interpreter's picks are farther away. [**ER**] chap2/. o3d-env-origfar,o3d-env-new-far



(a)



(b)

Figure 2.22: Segmentation results for another set of slices through the 3D cube, (a) before and (b) after interpreter guidance. The result is greatly improved, even far from where the manual salt picks were supplied. [**ER**] chap2/. 3d-origseg-far,3d-newseg-far

(NCIS) method. A single CPU performed the PRC segmentation in less than three minutes, while a full 3D segmentation of an image this size would not be feasible using the NCIS method, due to the complexity of the necessary eigenvector calculations. A direct comparison can be made, however, using a smaller 2D image such as the example in Figure 2.3(a). Segmenting this image using the PRC method as described here required less than one second, while the NCIS algorithm required 31 seconds using the same computational resources. Due to the differences in the cost functions of the NCIS and PRC algorithms, the computational advantage would be even greater for the PRC algorithm for larger and more realistic images.

Of course, computational efficiency means little if the resulting segmentation is not accurate. Again, a direct comparison of the relative accuracy of the NCIS and PRC methods can be obtained via analysis of the field data results. Figure 2.23 shows both calculated salt boundaries overlain on the image: the NCIS boundary is green/light-colored, and the PRC boundary is red/dark. 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 2.3(a) 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 may be a result of the denser network of graph edges created using the PRC method compared to the particular NCIS implementation shown here. This result serves to increase confidence in the viability of the PRC segmentation scheme as a useful interpretation tool.

CONCLUSIONS

Because of its global segmentation capabilities and extreme computational efficiency, the Pairwise Region Comparison (PRC) algorithm is an attractive candidate for seismic image segmentation. After modifications to the algorithm to account for the

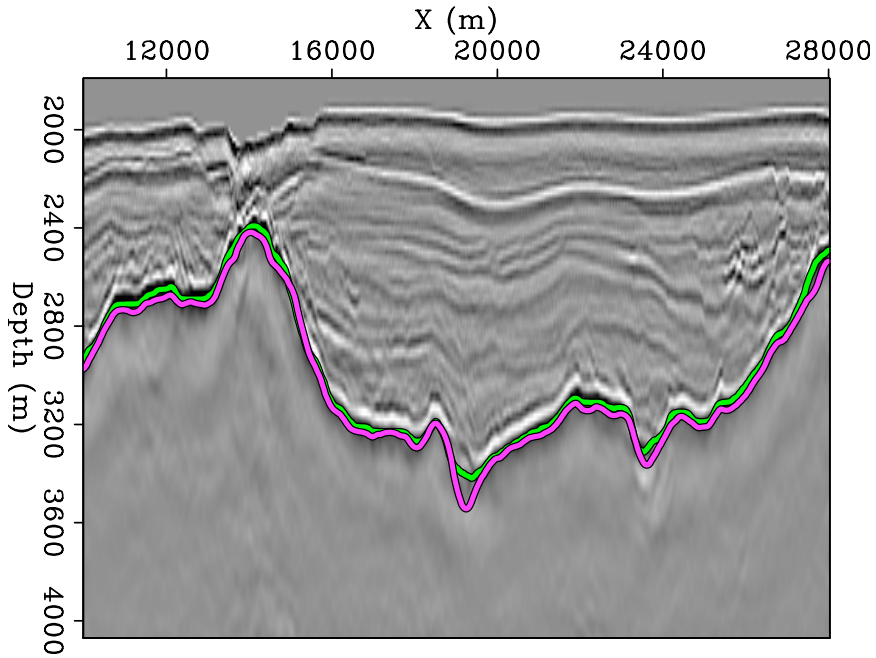


Figure 2.23: Salt boundary identified on a 2D image using two automatic segmentation techniques: NCIS (green/light color) and PRC (red/dark). [**NR**] chap2/. uno-segcomp

unique nature of seismic images, the PRC scheme successfully delineates salt structures on 2D and 3D seismic images. While fully automatic segmentations are sometimes successful, limited manual interpretations on one or more 2D slices can be used to guide a 3D, semi-automatic segmentation process. This allows for improved results throughout the image cube, not just near manual pick locations. The new algorithm performs extremely efficiently compared to other automatic interpretation techniques, and operates on the full seismic image or cube rather than a limited domain around a salt structure. These advantages could make it an important tool for interactive interpretation procedures and help streamline the model building workflow.

ACKNOWLEDGMENTS

I would like to thank Unocal/Chevron and Schlumberger Multiclient for providing the field datasets and images used here for examples, and SmaartJV for providing the Sigsbee synthetic model.