UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

SEISMIC DATA CONDITIONING, ATTRIBUTE ANALYSIS, AND MACHINE-LEARNING FACIES CLASSIFICATION: APPLICATIONS TO TEXAS PANHANDLE, AUSTRALIA, NEW ZEALAND, AND GULF OF MEXICO

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By
THANG HA
Norman, Oklahoma
2021

SEISMIC DATA CONDITIONING, ATTRIBUTE ANALYSIS, AND MACHINE-LEARNING FACIES CLASSIFICATION: APPLICATIONS TO TEXAS PANHANDLE, AUSTRALIA, NEW ZEALAND, AND GULF OF MEXICO

A DISSERTATION APPROVED FOR THE
SCHOOL OF GEOSCIENCES

BY THE COMMITTEE CONSISTING OF

To my mentors, all AASPI members, my friends, and my family

ACKNOWLEDGEMENTS

First, I want to express my deepest appreciation to Dr. Marfurt – my advisor, also my nicest “boss” of all time. It has been quite a long journey since I first joined Dr. Marfurt’s Attribute-Assisted Seismic Processing and Interpretation (AASPI) group in 2013, and I was lucky enough to constantly receive support and guidance from Dr. Marfurt, both academically and financially. I feel blessed that I was able to contribute back to the AASPI consortium with my coding skill while, at the same time, pursuing my PhD degree.

Second, I never forget all the advices and insights given to me by my committee members: Dr. Pranter, Dr. Chen, Dr. Devegowda, and, especially, Dr. Wallet – who offered me an internship related to my study.

Third, I am grateful for the financial supports and feedbacks from all AASPI sponsors, especially Dr. ICHIE Yasutaka from JX Nippon Company, Japan. I learned so much from the sponsors’ suggestions and comments, enabling me to improve the AASPI software.

Forth, I want to thank the “guardians” of our department: Ms. Rebecca Fay, Ms. Leah Moser, Ms. Ginger Leivas, and Ms. Ashley Tullius, for helping me keep track of my PhD progress and ease my financial burden. I also want to express my gratitude to my friends and colleagues, including (but not limited to) David Lubo-Robles, Saurabh Sinha, Dr. Rafael Pires De Lima, Dr. Sumit Verma, Dr. Tao Zhao, and Dr. Jie Qi, without whom I could not have gone so far in my field of research.

Finally, I want to thank my family – my wife Linh Vo and son Quy Ha, for staying with me, and my mom Thanh Tran, my dad Tuan Ha, and my sister Trang Ha, for supporting me mentally during my most struggling time as a PhD student.

FOREWORD

Data conditioning, seismic attributes, and machine learning have been extensively used in seismic interpretation, thanks to the evolution of computer hardware. Numerous researchers around the world have applied different data conditioning techniques, novel attributes, and cutting-edge machine learning algorithms to enhance the interpretation of specific geologic targets. However, as I attempted to apply some of these new techniques and algorithms to my research area, I usually got stuck where either there was no straightforward procedure to follow, or the mathematical formulation was not detailed enough to be reprogrammed by myself. Worse, I encountered many problems that were not mentioned in any available publication. Therefore, I carefully wrote down step-by-step workflows, re-derived complicated mathematical formulations from simpler equations, converted algorithms into pseudocodes, and documented all pitfalls and their workarounds (if any) for my research, in a hope that other students, researchers, and scientists can easily follow and repeat my work on their own datasets, without having to “learn the hard way” as I did.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Whether analyzed by a human interpreter or by a machine learning algorithm, 3D seismic interpretation is only as good as the data that goes into it. The goal of seismic processing is to minimize noise and enhance signal to provide the most accurate image of the subsurface. Once imaged, the resulting migrated data volume can be further enhanced to suppress random and cross-cutting coherent noise and to better balance the spectrum to improve vertical resolution. Next, seismic attributes enhance subtle geologic features that may be otherwise overlooked. At this point, skilled human interpreters are very adept at not only seeing patterns in the data, but also in constructing correlations in their brain between multiple attributes and geologic features of interest. Machine learning algorithms are not yet at this point. Several machine learning algorithms require, and many perform better on data that exhibit Gaussian statistics, such that we need to carefully scale the attribute volumes to be analyzed. The application of filters that block and smooth the attribute volume, mimicking what a human interpreter "sees" provide further improvements. In this dissertation, I address most of these data conditioning challenges, as well as adapting and recoding the machine learning algorithms themselves.

Conventional imaging of the shallow targets often results in severe migration aliasing. To improve the interpretation of a shallow fractured-basement reservoir in the Texas Panhandle, I developed a data conditioning technique called constrained conjugate-gradient least-squares migration to the prestack unmigrated data of the study area. I found that constrained conjugate-gradient least-squares migration can increase the signal-to-noise ratio, suppress migration artifacts, and improve seismic inversion results.

Although 3D seismic surveys are routinely acquired, in frontier areas, much of our data consist of a grid of 2D seismic lines. Few publications discuss the application and limitations of modern seismic attributes to 2D lines, and fewer still the application of machine learning. I used a grid of 2D lines acquired over a turbidite channel system and carbonate sequences in the Exmouth Plateau, North Carnarvon Basin, Australia, to address this question. First, I modified 3D data conditioning workflows including nonlinear spectral balancing and structure-oriented filtering, and found that spectral balancing followed by structure-oriented filtering provides superior results. All of the more common attributes perform well, but with analysis of 2D lines providing apparent dip and apparent curvature in the inline direction rather than true dip magnitude and azimuth, and most-positive and most-negative curvature and their strikes. I analyzed coherence, curvature, reflector convergence, and envelope attributes using self-organizing maps and was able to successfully map turbidite canyon, carbonate mounds, and mass-transport complexes (MTCs) in the study area.

Although some attributes exhibit Gaussian statistics, most do not. Although many machine learning algorithms are based on Gaussian statistics, most applications apply a simple Z-score normalization. I therefore compared the results of seismic facies classification of a Canterbury Basin turbidite system when using the traditional Z-score normalization versus one I developed that addresses skewness, kurtosis, and other scaling features in the attribute histogram. I found that logarithmic normalizations of skewed distributions are better input to unsupervised PCA, ICA, SOM, and GTM classification algorithms, but are worse for the supervised learning PNN classification algorithm. In contrast, supervised classification benefits greatly from a class-dependent normalization scheme, where the training data are normalized differently for each class.

CHAPTER 1: INTRODUCTION

Data conditioning, attribute analysis, and machine-learning classifications have been applied to 3D seismic data volume since they began to be routinely acquired in the 1990s. Since that time, the continued increase in computational power has enabled more sophisticated techniques of data conditioning and a long list of geometric, spectral, and texture attributes that provide improved input to an expanded list of unsupervised, semi-supervised and supervised machine learning techniques.

Seismic Data Conditioning

The goal of seismic data conditioning is to reduce noise and enhance features of interest in the original seismic data. Data conditioning can be as simple as removing abnormal data samples (also known as de-spiking) either by manual trace editing (e.g. Cahoj, 2015) or by using an absolute or statistical threshold (e.g. Ha and Kang, 2016). Another category of data conditioning is spectral balancing (or "whitening") the seismic data, which enhances the low and high frequencies of seismic data that may be less effectively excited and measured by the seismic experiment, attenuated due to scattering or absorption in the subsurface, or inadvertently suppressed through the seismic processing workflow (Chopra and Marfurt, 2016). More sophisticated data conditioning techniques include edge-preserving structure-oriented filtering (Hocker and Fehmers, 2002; Marfurt, 2007), which smooths noise and migration artifacts where the reflection events are continuous, while sharpening faults and discontinuities where they are not. Cabrales-Vargas (2011) performed prestack data conditioning by using a smoothing operator as a constraint for least-squares migration, in order to suppress migration operator aliasing. This prestack data conditioning workflow is further improved by Guo et al. (2016), in which the smoothing operator is replaced by a prestack structure-oriented filter. Data conditioning can be specific, targeting a particular type of noise as in the work of Verma et al. (2016), in which the authors suppressed coherent ground-roll by combining linear move-out correction with structure-oriented filtering. Most recently, Kim (2020) applied convolutional neural network in data conditioning by using a complex-valued residual convolutional neural network (ResNet) to predict seismic noise in the F-X domain.

Seismic Attribute Analysis

In simple terms, a seismic attribute is a result of applying mathematical algorithms to the original seismic amplitude data. The goal of seismic attributes is to enhance the visibility of specific geological features of interest that are otherwise hard to observe in the original amplitude data (Chopra and Marfurt, 2007). Different attributes are sensitive to different features and thus are suited for different interpretation purposes. The coherence and curvature family of attributes are routinely used to map discontinuities (faults, channel edges, erosional surfaces), curvilinear features (folds, channel axes, levees, carbonate mounds), as well as chaotic facies (salt, karst, mass-transport complexes) (Mai et al., 2009; Fisk et al., 2010; Chopra and Marfurt, 2011; Machado et al., 2015; Bhattacharya and Verma, 2019; Lyu et al., 2020). For stratigraphic interpretation, instantaneous attributes greatly help in picking layer terminations (Hardage, 1998), while spectral decomposition attributes can estimate layers' relative thicknesses and are sensitive to thin-bed tuning (Marfurt and Kirlin, 2001; Li et al., 2016). The impedance inversion family of attributes including P-impedance, S-impedance, v_P/v_S ratio, and density (Verma et al., 2013; Verma et al., 2018; Patel et al., 2019) integrate seismic data with well logs in order to estimate lithology, porosity, and geomechanical properties such as brittleness.

Machine-Learning Seismic Facies Classification

Among the first successful applications of machine learning to seismic facies classifications are the use of Kohonen self-organizing maps on seismic attributes for unsupervised facies analysis by Poupon et al. (1999) and Strecker and Uden (2002) and gas chimney semi-supervised classification using seismic attributes and neural networks by Meldahl et. al (1999). Since then, various machine-learning techniques have been adopted by seismic interpreters around the world, including (but not limited to) principal component analysis (Guo et al., 2009; Chopra and Marfurt, 2014), independent component analysis (Honorio et al., 2014; Lubo-Robles and Marfurt, 2019), k-means (Coleou et al, 2003; Zhao et al., 2015), Gaussian mixture model (Hardisty and Wallet, 2017), distance-preserving self-organizing maps (Zhao et al., 2016), generative topographic mapping (Zhao et al., 2015), proximal support vector machine (Zhao et al., 2014), random forest decision tree (Kim et al., 2019), and probabilistic neural network (Lubo-Robles et al., 2021).

The release of Google's Tensorflow in 2015 initiated rapid development of image-based classifications and pattern recognition in geological interpretation. Pires De Lima et al. (2020) applied convolutional neural network (CNN) to microscopic thin-section images to automatically classify fossils. Zhao (2018) applied an encoder-decoder CNN model to classify different seismic facies on an entire seismic line at once, rather than patch-based, partial CNN facies classification. Zhao (2019) continued these experiments using 3D CNN to design an automatic fault detection workflow without the need for a human interpreter to modify the training data (Wu et al., 2019; Zhao, 2019).

Insufficient Implementation Details

Despite a large number of existing publications showing promising results, there is still a lack of details and descriptions of problems encountered during the implementation of seismic data conditioning, attribute analysis, and machine-learning classifications. For example, Guo et al. (2016) successfully reduced seismic noise in a Mississippian limestone dataset using preconditioned least-squares prestack time migration. However, even with their mathematical description and a flow chart of their data conditioning workflow, I found it quite difficult to apply their workflow to my data due to the sheer complexity of least-squares migration. Hutchinson (2016) provided a workflow to interpret a 2D reconnaissance survey and recognized artifacts seen in the data, such as “wrap-ups” on a horizon picked from the 2D seismic lines, and glitches that appeared after data conditioning but did not identify their cause. Qi et al. (2020) acknowledged the importance of logarithmic data scaling to the input attributes used in their semi-supervised classification. However, they neither explained how they formulated their logarithmic transformation nor provided a side-by-side comparison between the traditional z-score normalization and their logarithmic transformation.

To address these shortcomings, I captured each step as I adopted these previous algorithms and workflows to my data volumes, providing details, showing intermediate results used in quality control, and where necessary, providing workarounds, with goal of making the work reproducible. My dissertation is structured as follows:

Although not mainstream because of the computational cost, least-squares migration has been successfully applied to marine data volumes by several of the larger geophysical service companies. The application of least-squares migration to the more sparsely sampled (aliased) land acquisition using Kirchhoff migration is much more challenging. In chapter 2, I converted the mathematical formulation of a constrained conjugate-gradient least-squares migration into a graphical user interface (GUI) that provides step-by-step instruction for each iteration in the workflow, keeping track of data domain at each step (i.e. migrated vs. unmigrated data), and providing the processor the ability to modify or even apply new constraints in the process. I then applied this workflow to the prestack unmigrated data of a fractured-basement play in the Texas Panhandle to improve the signal-to-noise ratio, suppress migration artifacts, and reduce noise in P-impedance seismic inversion result.

Complex trace seismic attributes were first applied to 2D grids of seismic data, but almost all subsequent developments, such as coherence, curvature, and GLCM textures, have been developed for 3D data volumes. Although 3D acquisition is now routine, large 2D grids of seismic data are still acquired and provide significant value in frontier areas, where “frontier” may mean offshore Greenland for an international oil company, and eastern Arkansas for a domestic shale resource play company. Modifying the algorithms to limit their application to 2D lines is relatively straightforward. Constructing software and GUIs to apply such attribute analysis to the entire grid of lines at once (rather than invoking hundreds of jobs, one for each line) took considerable effort. In chapter 3, I applied 3D seismic data conditioning, attribute analysis, and a machine-learning classification called self-organizing maps (SOM) to a 2D seismic survey covering about 40000 km^2 acquired over the in the Exmouth Plateau, North Carnarvon Basin, Australia. I improve the seismic image quality, interpret carbonate lithologies, perform facies classification on mass-transport complexes (MTCs), and identify potential targets including bright spots and carbonate mounds. I show that it is possible to interpret 2D seismic lines using modern 3D interpretation software packages, albeit not without compromises and workarounds. Azimuthal attributes, such as reflector convergence, must be extracted and interpolated component-by-component before being combined using trigonometric functions for a map view. I also find that the order of data conditioning steps is important to avoid introducing artifacts into the results. The display of the 3D grid also requires a workaround involving point-by-point extraction of SOM results in order to correctly display a regional MTC in a 3D viewport.

Several machine-learning algorithms, including k-means, Gaussian Mixture Models, Generative Topographic Maps, Probabilistic Neural Networks, and even Self-Organizing Maps, are built on an assumption of Gaussian statistics. Most commercial software packages use a simple z-score normalization. However, few attributes exhibit a Gaussian distribution. Although many attributes exhibit a log-normal distribution, several important attributes exhibit highly skewed and even relatively flat distributions. In chapter 4, I formulate a data adaptive logarithmic normalization scheme, using a suite of cascaded mathematical operations. I then apply this logarithmic normalization, along with the traditional z-score normalization, to the unsupervised classifications of a turbidite system in the Canterbury Basin, New Zealand, and a supervised classification of salt in the Eugene Island mini-basin, Gulf of Mexico. In general, a single normalization is applied to each attribute. However, for supervised classifications, I evaluate the performance of the classification where the normalization not only varies with each attribute but also depends on the facies (or class) being evaluated.

References

Bhattacharya, S. and S. Verma, 2019, Application of Volumetric Seismic Attributes for Complex Fault Network Characterization on the North Slope, Alaska: Journal of Natural Gas Science and Engineering, 65. doi: 10.1016/j.jngse.2019.02.002.

Cabrales-Vargas, A., 2011, Suppression of aliasing artifacts on 3D land data via constrained least-squares migration: Master's thesis, University of Oklahoma.

Cahoj, M., 2015, Reprocessing 3D seismic data for quantitative interpretation Forth Worth Basin, Jean, Texas: Master's thesis, University of Oklahoma.

Chopra, S. and K. J. Marfurt, 2007, Seismic Attributes for Prospect Identification and Reservoir Characterization: Society of Exploration Geophysicists. doi: 10.1190/1.9781560801900

Chopra, S. and K. J. Marfurt, 2011, Structural curvature versus amplitude curvature: The Leading Edge, 32, 980-984. doi: 10.1190/1.3628237

Chopra, S. and K. J. Marfurt, 2014, Churning seismic attributes with principal component analysis: SEG Expanded Abstract, 2672-2676. doi: 10.1190/segam2014-0235.1

Chopra, S. and K. J. Marfurt, 2016, Spectral decomposition and spectral balancing of seismic data: The Leading Edge, 35, 114-212. doi: 10.1190/tle35020176.1

Coleou, T., M. Poupon, and K. Azbel, 2003, Unsupervised seismic facies classification: A review and comparison of techniques and implementation: The Leading Edge, 22, 942–953. doi: 10.1190/1.1623635

Fangyu, L., J. Qi, and K. J. Marfurt, 2015, Attribute mapping of variable-thickness incised valley-fill systems: The Leading Edge, 34, 48-52. doi: 10.1190/tle34010048.1

Fisk, J., K. J. Marfurt, and D. Cooke, 2010, Correlating heterogeneous production to seismic curvature attributes in an Australian coalbed methane field: SEG Expanded Abstract. doi: 10.1190/1.3513316

Guo, S., B. Zhang, Q. Wang, A. Cabrales-Vargas, and K. J. Marfurt, 2016, Noise suppression using preconditioned least-squares prestack time migration: Application to the Mississippian Limestone: Journal of Geophysics and Engineering, **13**, 441-453.

Ha, T. and M. Kang, 2016, Arctic glacial deposits: A comprehensive 2D seismic processing and interpretation workflow: SEG Expanded Abstract. doi: 10.1190/segam2016-13778788.1

Hardage, B., V. M. Pendleton, J. L. Simmons, B. A. Stubbs, and B. J. Uszynski, 1998, 3-D Instantaneous Frequency Used as a Coherency/Continuity parameter To Interpret Reservoir Compartment Boundaries Across an Area of Complex Turbidite Deposition: Geophysics, 63, 1520-1531. doi: 10.1190/1.1444448

Hardisty, R. and B. Wallet, 2017, Unsupervised seismic facies from mixture models to highlight channel features: SEG Expanded Abstract, 2289-2293. doi: 10.1190/segam2017-17794438.1

Hocker, C., and G. Fehmers, 2002, Fast structural interpretation with structure-oriented filtering: The Leading Edge, 21, 238–243. doi: 10.1190/1.1463775.

Honorio, B. C. Z., A Crus Sanchetta, E. Pereira Leite, and A. Campane Vidal, 2014, Independent component spectral analysis: Interpretation, 2, SA21-SA29. doi: 10.1190/INT-2013-0074.1

Hutchinson, B., 2016, Application and Limitations of Seismic Attributes on 2D Reconnaissance Surveys: Master's thesis, University of Oklahoma.

Kim, Y., R. Hardisty, and K. J. Marfurt, 2019, Attribute selection in seismic facies classification: Application to a Gulf of Mexico 3D seismic survey and the Barnett Shale: Interpretation, 7, SE281-SE297. doi: 10.1190/INT-2018-0246.1

Kim, Y., 2020, Machine learning applications for seismic processing and interpretation: Doctoral dissertation, University of Oklahoma.

Li, F., S. Verma, H. Zhou, T. Zhao, and K. J. Marfurt, 2016, Seismic attenuation attributes with applications on conventional and unconventional reservoirs: Interpretation, **4**, SB63-SB77. doi: 10.1190/INT-2015-0105.1

Lubo-Robles, D. and K. J. Marfurt, 2019, Independent component analysis for reservoir geomorphology and unsupervised seismic facies classification in the Taranaki Basin, New Zealand: Interpretation, **7**, SE19-SE42. doi: 10.1190/INT-2018-0109.1

Lubo-Robles, D., T. Ha, S. Lakshmivarahan, K. J. Marfurt, and M. Pranter, 2021, Exhaustive Probabilistic Neural Network for attribute selection and supervised seismic facies classification: Interpretation, 9, doi: 10.1190/INT-2020-0102.1.

Lyu, B., J. Qi, F. Li, Y. Hu, T. Zhao, S. Verma, and K. J. Marfurt, 2020, Multispectral coherence: Which decomposition should we use? Interpretation, **8**, 1F-T215. doi: 10.1190/INT-2019-0124.1

Machado, G., A. Alali, B. Hutchinson, O. Oluwatobi, and K. J. Marfurt, 2015, Improving fault images using a directional Laplacian of a Gaussian operator: SEG Expanded Abstract. doi: 10.1190/segam2015-5920781.1

Mai, H., K. J. Marfurt, and S. Chavez-Perez, 2009, Coherence and volumetric curvatures and their spatial relationship to faults and folds, an example from Chicontepec Basin, Mexico: SEG Expanded Abstract. doi: 10.1190/1.3255033

Marfurt, K. and R. Kirlin, 2001, Narrow-band spectral analysis and thin-bed tuning, Geophysics, 66, 988-1328. doi: 10.1190/1.1487075

Marfurt, K., 2018, Seismic Attributes as the Framework for Data Integration Throughout the Oilfield Life Cycle: Society of Exploration Geophysicists. doi: 10.1190/1.9781560803522

Meldahl, P., R. Heggland, B. Bril, and P. de Groot, 1999, The chimney cube, an example of semi-automated detection of seismic objects by directive attributes and neural networks: Part I; methodology: SEG Expanded Abstract, 931-934. doi: 10.1190/1.1821262

Patel, S., F. Oyebanji, and K. J. Marfurt, 2019, Compensating for migration stretch to improve the resolution of S-impedance and density inversion: SEG Expanded Abstract, 630-634. doi: 10.1190/segam2019-3215872.1

Pires De Lima, R., K. F. Welch, J. E. Barrick, K. J. Marfurt, R. Burkhalter, M. Cassel, and G. S. Soreghan, 2020, Convolutional neural networks as an aid to biostratigraphy and micro-paleontology: a test on late Paleozoic microfossils: Palaios, **35**, 391-402. doi: 10.2110/palo.2019.102

Poupon, M., K. Azbel, and G. Palmer, 1999, A new methodology based on seismic facies analysis and litho-seismic modeling The Elkhorn Slough field pilot project, Solano County, California: SEG Expanded Abstract, 927-930. doi: 10.1190/1.1821260

Qi, J., B. Zhang, B. Lyu, and K. J. Marfurt, 2020, Seismic attribute selection for machine-learning-based facies analysis: Geophysics, **85**, O17-O35. doi: 10.1190/GEO2019-0223.1

Strecker, U., and R. Uden, 2002, Data mining of 3D poststack attribute volumes using Kohonen self-organizing maps: The Leading Edge, **21**, 1032-1037. doi: 10.1190/1.1518442

Verma, S., Y. D. Moro, and K. J. Marfurt, 2013, Pitfalls in prestack inversion of merged seismic surveys: Interpretation, 1, A1–A9. doi: 10.1190/INT-2013-0024.1

Verma, S., S. Guo, T. Ha, and K. J. Marfurt, 2016, Highly aliased ground-roll suppression using a 3D multiwindow Karhunen-Loeve filter: Application to a legacy Mississippi Lime survey: Geophysics, **81**, V79-V88.

Verma, S., S. Bhattacharya, B. Lujan, D. Agrawal, and S. Mallick, 2018, Delineation of early Jurassic aged sand dunes and paleo-wind direction in southwestern Wyoming using seismic attributes, inversion, and petrophysical model: Journal of Natural Gas Science and Engineering, 60, 1-10. doi: 10.1016/j.jngse.2018.09.022

Wu, X., L. Liang, Y. Shi, and S. Fomel, 2019, FaultSeg3D: Using synthetic data sets to train an end-to-end convolutional neural network for 3D seismic fault segmentation: Geophysics, **84**, ix-Z16. doi: 10.1190/geo2018-0646.1

Zhao, T., V. Jayaram, and K. J. Marfurt, 2014, Lithofacies classification in Barnett Shale using proximal support vector machines: SEG Expanded Abstract. doi: 10.1190/segam2014-1210.1

Zhao, T., V. Jayaram, A. Roy, and K. J. Marfurt, 2015, A comparison of classification techniques for seismic facies recognition: Interpretation, 3, SAE29–SAE58. doi: 10.1190/INT-2015-0044.1.

Zhao, T., J. Zhang, F. Li, and K. J. Marfurt, 2016, Characterizing a turbidite system in Canterbury Basin, New Zealand, using seismic attributes and distance-preserving self-organizing maps: Interpretation, 4, SB79-SB89. doi: 10.1190/INT-2015-0094.1.

Zhao, T., 2018, Seismic facies classification using different deep convolutional neural networks: SEG Expanded Abstract. doi: 10.1190/segam2018-2997085.1

Zhao, T., 2019, 3D convolutional neural networks for efficient fault detection and orientation estimation: SEG Expanded Abstract. doi: 10.1190/segam2019-3216307.1

CHAPTER 2: THE VALUE OF CONSTRAINED CONJUGATE-GRADIENT LEAST-SQUARES MIGRATION IN SEISMIC INVERSION: APPLICATION TO A FRACTURED-BASEMENT PLAY, TEXAS PANHANDLE

Abstract

Seismic inversion has become almost routine in quantitative 3D seismic interpretation. To ensure the quality of the seismic inversion, the input seismic data need to have a high signal-to-noise ratio. With the current low oil price environment, seismic reprocessing is often preferred over reacquisition to improve the data quality. Common filter pairs include forward and inverse f-k and Radon transforms. Forward and inverse migrations (i.e. migration and demigration) are a more recently introduced transform pair that, when used together in an iterative workflow, results in a least-squares migration algorithm. Least-squares migration compensates for surface variation in data density and, when combined with a filter applied to the prestack migrated images, suppresses both operator and data aliasing. I apply a least-squares migration workflow to a fractured-basement dataset from Texas Panhandle to demonstrate the enhancement in signal-to-noise ratio, the reduction in acquisition footprint and migration artifacts, and the improvement in the P-impedance inversion result.

Introduction

Thanks to improved algorithms and simple user interfaces, seismic impedance inversion has become a routine means of incorporating well data to estimate lithological properties in 3D quantitative interpretation. However, the quality of seismic inversion depends on the quality of the seismic amplitude data. To improve seismic data quality, we either acquire and process new data, or reprocess the old data to obtain a better, more amplitude-friendly image. While reacquiring data using denser, wider azimuth survey is more likely to produce better images, such acquisition is both costly and time consuming. Given the current low oil price, reprocessing might be the only feasible choice for many operators.

Reprocessing old seismic data involves many different tasks, including surface-consistent residual statics correction, velocity model refinement, coherent noise suppression, trace balancing, 5D interpolation, prestack time/depth migration, and other forms of data conditioning. Each of these tasks contributes to the final image improvement. In this chapter, I focus on the last two tasks – migration and data conditioning through the use of constrained conjugate-gradient least-squares migration method.

Nemeth et al. (1999) were perhaps the first to use migration and demigration (least-squares migration) as a seismic processing pair with Nemeth et al. (2000) showing how one can separate signal from noise. Most subsequent least-squares migration has focused on marine data, where the acquisition, while aliased, is still regular, resulting in Yu et al's (2006) migration deconvolution algorithm. Since then, the use of least-squares migration has further advanced, with modern implementations by Zeng (2014) inverting for impedance changes rather than reflectivity at offset. Applications of least-squares migration of 3D land data have been much more limited, with Guo et al. (2016) applying constrained least-squares migration to Mississippi

Lime plays in Ness Co., KS and Osage Co., OK, and Verma et al. (2016) applying constrained least-squares migration to a Mississippi Lime survey in north Texas. My work builds on these last two publications.

I begin this chapter with a short review of the least-squares migration workflow, with mathematical and workflow implementation relegated to the Appendices. I then describe the data quality and processing challenges for a 3D data volume acquired over fractured basement in the Texas Panhandle. Although this is a modern wide-azimuth survey acquired in 2013 with a nominal bin size of 82.5 by 82.5 ft, the shallow target at 2500 ft results in strong operator aliasing and acquisition footprint. I apply conventional migration and constrained conjugate-gradient least-squares migration to this data volume and compare the results. Finally, I validate my findings through seismic attributes, improved well ties, and P-wave impedance inversion.

Method

Migration is central to seismic imaging. The main idea behind migration is to broadcast each amplitude sample onto an ellipsoidal pattern computed from the velocity model and then stack all the resulting ellipsoids (Figure 2.1). If adequately sampled on the surface and given an accurate velocity model, areas of constructive interference result in reflectors and diffractors. Other areas of destructive interference result in low reflectivity. Compared to simple stacked images of NMO-corrected gathers, migration collapses diffractors and moves (or “migrates”) dipping reflectors to steeper dips in the updip direction.

If the surface data are insufficiently sampled, the image suffers from two types of aliasing. First, the ellipsoids associated with reflected and diffracted energy may not destructively interfere at steep dips, giving rise to operator aliasing (Figure 2.1). Second, undersampled noise such as ground roll and shallow diffractions, which (if properly sampled) should exhibit short apparent wavelength and strong moveout and should be filtered out by migration, are instead aliased to longer apparent wavelength and gentler moveout and will be passed and subsequently imaged by migration (Pramik, 2011; Dev and McMechan, 2009). Biondi (2001) describes a common workflow to suppress operator aliasing that simply filters out higher frequencies when migrating to steeper dips.

The key objectives of my least-squares migration workflow are thus to (1) suppress aliased signal and aliased noise in the final image, (2) preserve rather than reject the higher frequency information of steeply dipping reflectors (Zeng et al, 2017), and (3) compensate for irregular surface sampling that gives rise to acquisition footprint and other amplitude artifacts.

To understand constrained conjugate-gradient least-squares migration, let me break down the method into four elements: migration, least-squares, conjugate-gradient, and constraint. The meaning of each element is discussed below, in that order, with details given in the appendices.

Migration

The migration operator can be understood as a filter implemented as a matrix operator, applied to the prestack data to produce a reflectivity model (i.e. the migrated images). The reverse operator is demigration (more commonly known as the forward modeling operator). This demigration operator is usually denoted as **G** (for Greens' function), which is a filter that, when applied to a reflectivity model **m**, would produce the prestack raw seismic data **d**:

$$\mathbf{d} = \mathbf{Gm}. \tag{2.1}$$

I want to solve for the reflectivity model **m**. To do so, I need to invert operator **G**:

$$\mathbf{m} = \mathbf{G}^{-1}\mathbf{d}. \tag{2.2}$$

However, in almost all cases, the reflectivity model **m** and the prestack raw seismic data **d** do not have the same dimensional configuration, and therefore the demigration operator **G** is not only a non-square matrix but may also be "rank deficient", which physically means some areas of the subsurface are poorly illuminated (such as shadow zone in Kirchhoff migration). Such a matrix **G** cannot be directly inverted. Fortunately, the inverse of **G** can be approximated by its transpose, $\mathbf{G}^\mathbf{T}$ (Nemeth et al, 2000). Hence, migration is simply the transpose of demigration:

$$\mathbf{m} = \mathbf{G}^\mathbf{T}\mathbf{d}. \tag{2.3}$$

Using $\mathbf{G}^\mathbf{T}$ as an approximation of $\mathbf{G}^{-1}$ typically produces adequate image quality for dense data. However, for old data with low fold or modern data like my application with shallow target, this approximation no longer holds, resulting in the annoying cross-cutting migration artifacts seen on the final image.

Least-squares

The goal of "least-squares" is to find a solution **m** such that the *sum of the squares* of all elements of the error (**d-Gm**) is the *least*. That is, instead of finding **m** such that

$$\mathbf{d}\text{-}\mathbf{Gm} = \mathbf{0}, \tag{2.4}$$

I want to find **m** such that the objective function

$$J=(\mathbf{d}\text{-}\mathbf{Gm})^T(\mathbf{d}\text{-}\mathbf{Gm}) \tag{2.5}$$

is minimum.

Conjugate-Gradient

The conjugate-gradient method was first developed by Hestenes and Stiefe (1952). For a least-squares problem in the form:

$$\mathbf{Ax}=\mathbf{b}, \tag{2.6}$$

The conjugate-gradient method involves finding a set of **A**-conjugate directional vectors $\mathbf{S}=\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}\}$ and incorporating them to the solution **x** iteratively. Appendix B explains the conjugate-gradient method in detail.

The key advantage that differentiates the conjugate-gradient method from other first-order iterative approaches to solve least-squares problems, including QR-decomposition, singular value decomposition (SVD), and steepest decent method, is that theoretically, it guarantees to converge after *n* iterations for an *n*-by-*n* matrix **A**, while other methods cannot guarantee to converge after a finite number of iterations (Lewis et al, 2006). Therefore, the conjugate-gradient method generally has the fastest convergence rate – the rate at which we approach the true solution iteratively.

Constraints

Last but not least, keep in mind that the data consists of both signal *and* noise. Because of this, if I only focus on solving the exact solution for the least-squares problem, I am in fact solving for a reflectivity model that is responsible for both signal *and* noise. I do not want my seismic image to represent the noise in the raw gathers. Therefore, I need to constrain the reflectivity model to only represent the signal portion of the data. The constraint I use in my least-squares migration workflow is the prestack structure-oriented filtering (SOF). This constraint involves calculating structural dip from the stacked volume, computing coherence attribute along structural dip, and finally applying lower-upper-middle (LUM) filter to the input data based on coherency. Data are filtered where coherency is high (e.g. strong reflectors) and are kept the same where coherency is low (e.g. faults and discontinuities). Therefore, SOF suppresses random noise and aliasing artifacts that cuts across the dominant reflectors, while preserving edges (Zhang et al, 2016). My workflow follows that of Guo et al (2016), while appendix E provides details on my implementation of these constraints within the conjugate-gradient solution framework.

Application and Results

I applied the constrained conjugate-gradient least-squares migration to a Texas Panhandle dataset. The Panhandle-Hugoton field, of Texas, Oklahoma, and Kansas, is a giant oil field and the largest conventional gas field in North America, with an estimated ultimate recovery (EUR) of 1400 million barrels of oil and 75 trillion cubic feet of gas (Sorenson, 2005). Although the field has been extensively produced, previously untapped local hydrocarbon accumulations are still encountered. Recent drilling activity indicates that some wells produce directly from basement fractures, suggesting a shallow “buried-hill” reservoir type (Figure 2.2). My main objective is to use seismic attributes and inversion results to identify open fracture zones that are potentially filled with hydrocarbon.

The top basement is very shallow (~2500-ft-deep, equivalent to ~600 ms two-way-travel time), giving rise to some processing challenges. Overlaying on top of the basement is a thick Permian evaporite layer, causing strong head waves (Figure 2.3). Reflection signals are overprinted by strong coherent noise, including ground roll and reverberating refractions (Figure 2.4). Due to the shallow target, some gaps in the source-receiver geometry, and the nature of the orthogonal shot and receiver line acquisition program (Figure 2.5), the seismic data suffer from acquisition footprint.

After applying coherent noise suppression techniques described by Verma et al (2016), I evaluated both conventional Kirchhoff prestack time migration and constrained conjugate-gradient least-squares migration. I then computed geometric attributes and impedance inversion from both volumes in order to quantify any improvement from constrained conjugate-gradient least-squares migration.

Figure 2.6 shows a vertical slice through the seismic amplitude volumes generated by conventional Kirchhoff, unconstrained conjugate-gradient least-squares, and constrained conjugate-gradient least-squares prestack time migration. The conventional Kirchhoff migrated result exhibits strong steeply dipping migration artifacts due to operator aliasing, even though I used only the low-frequency components to image steep dips as described by Biondi (2001). This aliasing gives rise to acquisition footprint in subsequent attribute and inversion results. Without the constraint, least-squares migration image enhances those artifacts because the least-squares element alone preserves noise components that have leaked through migration. The constrained conjugate-gradient least-squares migration result increases the signal-to-noise ratio, enhances reflection clarity, and fills in the illumination gaps caused by highways. Both SOF constraint and least-squares migration contributes to a higher signal-to-noise ratio and a better amplitude balancing. The SOF constraint takes an initial reflectivity model and rejects components that are inconsistent with its neighbors and with the local dips. Least-squares migration then adjust this model further to better fit the (sparse) surface data.

Figure 2.7 shows coherence time slices below the top basement generated by conventional Kirchhoff and constrained conjugate gradient least-squares prestack time migration. Most of the grid-like (hash) artifacts are suppressed on the coherence time slice computed from the constrained conjugate-gradient least-squares migration result.

The same effect can be observed in the near-offset-stack P-impedance inversion results and inversion misfit error maps (Figure 2.8 and Figure 2.9). Acquisition footprint is greatly reduced in the constrained conjugate-gradient least-squares migration result, making low impedance zones of interest smoother and easier to identify. These low impedance zones contain producing well locations in the survey area and correspond to open fractures filled with hydrocarbons. The impact on prestack azimuthal anisotropy analysis is also significant (Ha, 2017). Unfortunately, the absence of an S-wave sonic log prevented prestack inversion.

Conclusions

Evaluation of the results of the constrained conjugate-gradient least-squares migration on a Texas Panhandle dataset shows a significant improvement in seismic data quality by reducing migration artifacts, suppressing acquisition footprint, and enhancing reflection clarity. Zones of low impedance, hydrocarbon-filled open fractures are better delineated using inversion results from the constrained conjugate-gradient least-squares migration. Three iterations of the conjugate-gradient solution required three prestack migrations and three prestack demigrations, with a computation cost six times that of the conventional migration. This increase in computation cost is small compared to the velocity analysis cost associated with conventional migration followed by residual velocity analysis in a Deregowski loop. In contrast, the same velocity is used for each iteration of migration and demigration. My caveat is that interpreters must recognize that even this sophisticated workflow needs careful data processing. Specifically, a good velocity model, surface consistent residual statics corrections, and coherent noise suppression, are equally important. Only together with a good velocity model and higher signal-to-noise data can constrained conjugate-gradient least-squares migration exhibit its full potential.

Acknowledgments

I thank Schlumberger for a license to their Vista software package used in preprocessing the data and CGG Veritas for Hampson-Russell used in inversion, and all of the sponsors of our AASPI consortium. Thanks to CIMAREX Energy Company for providing me a license to their seismic data volume for use in research and education. Financial support was through the industry sponsors of the OU Attribute-Assisted Seismic Processing and Interpretation (AASPI) consortium.

Appendix A: Least-Squares Migration

In this section, I provide the mathematical background for least-squares migration, starting with the well-known linear equation:

$$\mathbf{d} = \mathbf{Gm} \tag{2.A-1}$$

where

d is the original prestack data measured on the earth's surface,

G is the forward Kirchhoff modeling operator (i.e. the "demigration" operator), and

m is the true migrated result, which is what we seek.

In general, **G** is not a square matrix, because **d** and **m** may have different lengths. Therefore, **G** cannot be simply inverted and put on the other side of the equation. The transpose of **G** (demigration) operator, $\mathbf{G}^\mathbf{T}$, is the Kirchhoff migration operator. Usually,

$$\mathbf{m}' = \mathbf{G}^\mathbf{T}\mathbf{d} \tag{2.A-2}$$

is considered as the migrated result, which is an accurate assumption if the surface data are both regularly and densely sampled. Typically, the scale of the migrated result is not equivalent to the scale of the original data, although relative amplitude changes, such as the AVO and AVAz effects, are preserved. $\mathbf{G}^\mathbf{T}$ (migration operator) is a good approximation to the inverse of **G** (demigration operator). This assumption is not valid under the mathematical lens, particularly for undersampled or irregularly sampled data.

Instead, let us solve for **m** that minimize the following objective function:

$$J = ||\mathbf{d}\text{-}\mathbf{Gm}||^2 \tag{2.A-3}$$

Expanding J, I obtain

$$J = ||\mathbf{d}\text{-}\mathbf{Gm}||^2 = (\mathbf{d}\text{-}\mathbf{Gm})^\mathbf{T}(\mathbf{d}\text{-}\mathbf{Gm}) = \mathbf{d}^\mathbf{T}\mathbf{d}\text{-}(\mathbf{Gm})^\mathbf{T}\mathbf{d}\text{-}\mathbf{d}^\mathbf{T}(\mathbf{Gm})+(\mathbf{Gm})^\mathbf{T}(\mathbf{Gm})$$

$$= \mathbf{d}^\mathbf{T}\mathbf{d}\text{-}\mathbf{m}^\mathbf{T}\mathbf{G}^\mathbf{T}\mathbf{d}\text{-}\mathbf{d}^\mathbf{T}\mathbf{Gm}+\mathbf{m}^\mathbf{T}(\mathbf{G}^\mathbf{T}\mathbf{G})\mathbf{m} = \mathbf{d}^\mathbf{T}\mathbf{d}\text{-}2\mathbf{d}^\mathbf{T}\mathbf{Gm}+\mathbf{m}^\mathbf{T}(\mathbf{G}^\mathbf{T}\mathbf{G})\mathbf{m} \tag{2.A-4}$$

where the scalars

$\mathbf{m^T G^T d = d^T G m}$ (2.A-5)

To find the minimum of *J*, I take the gradient of *J* (∇J) and set it to zero:

∇J $\mathbf{= -2G^T d + 2G^T G m} = 0$, or (2.A-6)

$\mathbf{G^T G m = G^T d}$ (2.A-7)

An alternative way to write equation 2.A-7 is to turn equation 2.A-1 into a “normal” equation by multiplying both sides with the transpose of **G** (i.e. $\mathbf{G^T}$). From here, I can multiply both side by the inverse of $\mathbf{G^T G}$ (i.e. $(\mathbf{G^T G})^{-1}$) to obtain

$(\mathbf{G^T G})^{\mathbf{-1}}\mathbf{G^T G m} = (\mathbf{G^T G})^{\mathbf{-1}}\mathbf{G^T d}$ (2.A-8)

such that the left of m becomes an identity matrix, giving

$\mathbf{m} = (\mathbf{G^T G})^{\mathbf{-1}}\mathbf{G^T d}$. (2.A-9)

To avoid instability, least-squares solution (such as in deconvolution) often introduces a pre-whitening factor $(\varepsilon\mathbf{I})$, which is a fraction of the diagonal of $\mathbf{G^T G}$ to obtain

$\mathbf{m} = (\mathbf{G^T G} + \varepsilon\mathbf{I})^{\mathbf{-1}}\mathbf{G^T d}$ (2.A-10)

thereby favoring the solution with the minimum reflectivity energy. In my application, I use a constraint (SOF) that favors piece-wise continuous (i.e. edge-preserving) solutions over all other solutions.

Appendix B: Conjugate-Gradient Least-Squares Migration

The linear system involved in the conjugate gradient method has the form

$\mathbf{b} = \mathbf{Ax}$, (2.B-1)

where

$\mathbf{A}=\mathbf{G}^{\mathbf{T}}\mathbf{G}$, (2.B-2)

$\mathbf{x}=\mathbf{m}$, and (2.B-3)

$\mathbf{b}=\mathbf{G}^{\mathbf{T}}\mathbf{d}$. (2.B-4)

To understand the conjugate *gradient* method, I first examine the conjugate *direction* method. A set of vectors $\mathbf{S}=\{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{n-1}\}$ is said to be **A**-conjugate if $\mathbf{p}_i^{\mathbf{T}}\mathbf{A}\mathbf{p}_j=0$ for $i \neq j$ ($i, j \in [0,n-1]$). If such a set of vectors is provided, the conjugate *direction* method is guaranteed to converge after *n* iterations. A summary of the conjugate *direction* method is as follows:

```
Step 0: Choose a starting point $\mathbf{x_0}$. Compute $\mathbf{r_0}=\mathbf{b}-\mathbf{Ax_0}$.

For $k=0$ to $(n-1)$ do:

- Step 1: $\boldsymbol{\alpha_k} = \frac{\mathbf{p_k^T r_k}}{\mathbf{p_k^T A p_k}}$

- Step 2: $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$

- Step 3: $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{A}\mathbf{p}_k$

- Step 4: If the residual $\|\mathbf{r}_{k+1}\| < \varepsilon$, a convergence testing threshold, then $\mathbf{x}_{k+1}$ is the solution,
  and one quits the loop. Otherwise, continue.
```

Normally the set of vector $\mathbf{p}_i$ is not known beforehand. Instead, a specific set of vectors $\mathbf{p}_i$ is generated while iterating through the conjugate *gradient* method:

Step 0: Choose a starting point $\mathbf{x_0}$. Compute $\mathbf{r_0}=\mathbf{b}-\mathbf{A}\mathbf{x_0}$. Let $\mathbf{p_0}=\mathbf{r_0}$

For $k=0$ to $(n-1)$ do:

- Step 1: $\boldsymbol{\alpha_k} = \frac{\mathbf{r_k^T r_k}}{\mathbf{p_k^T A p_k}}$

- Step 2: $\mathbf{x_{k+1}} = \mathbf{x_k} + \alpha_k \mathbf{p_k}$

- Step 3: $\mathbf{r_{k+1}} = \mathbf{r_k} - \alpha_k \mathbf{A p_k}$

- Step 4: If the residual $||\mathbf{r_{k+1}}|| < \varepsilon$, a convergence testing threshold, then $\mathbf{x_{k+1}}$ is the solution, and one quits the loop. Otherwise, continue.

- Step 5: $\boldsymbol{\beta_k} = \frac{\mathbf{r_{k+1}^T r_{k+1}}}{\mathbf{r_k^T r_k}}$

- Step 6: $\mathbf{p_{k+1}} = \mathbf{r_{k+1}} + \boldsymbol{\beta}_k \mathbf{p_k}$

Among the various methods to solve the least-squares problem, the conjugate gradient method usually yields the fastest rate of convergence (the algorithm does not requires external input and guarantees to converge after *n* iterations). For these reasons, it is often the preferred method in many fields of study, including migration in geophysics. For consistency with the notation used in previous geophysical research, I replace **p** with **h**, and **r** with **g**:

Step 0: Choose a starting solution $\mathbf{m_0}$. Compute $\mathbf{g_0}=\mathbf{G^T d} - \mathbf{G^T G m_0}$. Let $\mathbf{h_0}=\mathbf{g_0}$ be the conjugate direction.

For $k=0$ to $(n-1)$ do:

- Step 1: $\boldsymbol{\alpha_k} = \frac{\mathbf{g_k^T g_k}}{\mathbf{h_k^T (G^T G) h_k}} = \frac{\mathbf{g_k^T g_k}}{\mathbf{(G h_k)^T G h_k}}$

- Step 2: $\mathbf{m_{k+1}} = \mathbf{m_k} + \alpha_k \mathbf{h_k}$

- Step 3: $\mathbf{g_{k+1}} = \mathbf{g_k} - \alpha_k \mathbf{G^T G h_k}$

- Step 4: If $||\mathbf{g_{k+1}}|| < \varepsilon$, a convergence testing threshold, then $\mathbf{m_{k+1}}$ is the solution, and one quits the loop. Otherwise, continue.

- Step 5: $\boldsymbol{\beta_k} = \frac{\mathbf{g_{k+1}^T g_{k+1}}}{\mathbf{g_k^T g_k}}$

- Step 6: $\mathbf{h_{k+1}} = \mathbf{g_{k+1}} + \boldsymbol{\beta}_k \mathbf{h_k}$

Because migration ($\mathbf{G^T}$ is the migration operator) and demigration ($\mathbf{G}$ is the demigration operator) are computationally intensive, I want to minimize the number of times I apply them. Therefore, I introduce **r** as the residual in the demigrated domain (i.e. the domain of the original data **d**), in contrast to **g** which is the residual in the model domain (i.e. the domain of the solution **m**).

To do so, in step 0, set

$\mathbf{r_0} = \mathbf{d} - \mathbf{G m_0}$, and (2.B-5)

$\mathbf{g_0} = \mathbf{G^T r_0} = \mathbf{G^T (d - G m_0)} = \mathbf{G^T d - G^T G m_0}$. (2.B-6)

Also, to make the result unbiased, I choose

$\mathbf{m_0} = \mathbf{0}$. (2.B-7)

Thus,

$\mathbf{r_0} = \mathbf{d}$ and (2.B-8)

$\mathbf{g_0} = \mathbf{G^T d}$. (2.B-9)

In step 1: note how $\mathbf{A}=\mathbf{G}^\mathbf{T}\mathbf{G}$ changes the denominator to a simple dot product. I only need to apply demigration $\mathbf{G}$ to the conjugate gradient $\mathbf{h_k}$ and save one migration operator $\mathbf{G}^\mathbf{T}$.

In step 3: note that $\mathbf{g_{k+1}}$ is the migrated $\mathbf{r_{k+1}}$, and $\mathbf{g_k}$ is the migrated residual $\mathbf{r_k}$ (i.e. $\mathbf{g_{k+1}}=\mathbf{G}^\mathbf{T}\mathbf{r_{k+1}}$, and $\mathbf{g_k}=\mathbf{G}^\mathbf{T}\mathbf{r_k}$). Therefore, I can separate step 3 into two parts:

$\mathbf{r_{k+1}} = \mathbf{r_k} - \alpha_k\mathbf{Gh_k}$, and (2.B-10)

$\mathbf{g_{k+1}} = \mathbf{G}^\mathbf{T}\mathbf{r_{k+1}}$. (2.B-11)

Since step 1 (calculating α_k) involves reading data from $\mathbf{g_k}$ and $\mathbf{Gh_k}$, it would be more efficient to merge step 1 with $\mathbf{r_{k+1}}=\mathbf{r_k}-\alpha_k\mathbf{Gh_k}$. Similarly, step 5 and step 6 can be merged for the same reason.

Applying the above modifications, I can rewrite the conjugate gradient method for migration as follow:

Step 0: Choose a starting solution $\mathbf{m_0}=\mathbf{0}$. Set $\mathbf{r_0}=\mathbf{d}$. Compute $\mathbf{g_0}=\mathbf{G}^\mathbf{T}\mathbf{d}$. Let $\mathbf{h_0}=\mathbf{g_0}$ be the conjugate direction.

For $k=0$ to (n-1) do:

- Step 1: $\boldsymbol{\alpha_k} = \frac{\mathbf{g_k^T g_k}}{(\mathbf{Gh_k})^\mathbf{T}\mathbf{Gh_k}}$, and $\mathbf{r_{k+1}}=\mathbf{r_k} - \alpha_k\mathbf{Gh_k}$
- Step 2: $\mathbf{m_{k+1}} = \mathbf{m_k}+\alpha_k\mathbf{h_k}$
- Step 3: $\mathbf{g_{k+1}} = \mathbf{G}^\mathbf{T}\mathbf{r_{k+1}}$
- Step 4: If $||\mathbf{g_{k+1}}|| < \varepsilon$, a convergence testing threshold, then $\mathbf{m_{k+1}}$ is the solution, and one quits the loop. Otherwise, continue.
- Step 5: $\boldsymbol{\beta_k} = \frac{\mathbf{g_{k+1}^T g_{k+1}}}{\mathbf{g_k^T g_k}}$, and $\mathbf{h_{k+1}} = \mathbf{g_{k+1}} + \boldsymbol{\beta}_k\mathbf{h_k}$

The above suit of equations should be familiar to most people working with conjugate gradient least-squares migration. What I have done here is nothing more than a “translation” between mathematical papers and geophysical migration, simply by replacing notations of variables.

Appendix C: Constrained Least-Squares Migration

The next step is to add a constraint to the least-squares migration workflow. The purpose of the constraint is to increase the signal-to-noise ratio and reduce migration aliasing artifacts. A constraint is simply a filter **F** applied to the solution **m** – a means to "bend" the result to my will:

$$\bar{\mathbf{m}}_{\mathbf{k+1}} = \mathbf{F}(\mathbf{m}_{\mathbf{k+1}}) = \mathbf{F}(\mathbf{m}_{\mathbf{k}} + \boldsymbol{\alpha}_{k}\mathbf{h}_{\mathbf{k}}). \quad (2.C\text{-}1)$$

To improve the conjugate direction, I also need to apply filter **F** to $\mathbf{h}_\mathbf{k}$: $\bar{\mathbf{h}}_{\mathbf{k}} = \mathbf{F}(\mathbf{h}_{\mathbf{k}})$

Because the filter **F** might be computationally intensive, **F** is usually assumed to be a relatively linear operator. Therefore, I can reduce computational cost by setting $\bar{\mathbf{h}}_{\mathbf{k}} = \frac{\bar{\mathbf{m}}_{\mathbf{k+1}} - \mathbf{m}_{\mathbf{k}}}{\boldsymbol{\alpha}_{k}}$

As I introduce the constraint to the workflow, I expect the residual $\mathbf{r}_\mathbf{k}$ to be as close to the noise portion of the input data as possible. That is, the magnitude of the residual, $||\mathbf{r}_{\mathbf{k}}||$, should become smaller and then stabilize, but never reach zero. Since I do not know the magnitude of the noise portion in the original data, a constant threshold ε of the residual's magnitude is no longer valid as a stopping criterion. I need a new condition to quit the loop.

In all of my experiments, the result is considered sufficiently improved for interpretation purposes when the change of the residual is within 10% of the residual in the current iteration (typcially after three iterations). Therefore, I define a new stopping criterion for my workflow as when $\frac{||\mathbf{r}_{\mathbf{k+1}} - \mathbf{r}_{\mathbf{k}}||}{||\mathbf{r}_{\mathbf{k+1}}||} < 0.1$.

The constrained conjugate gradient least-squares migration workflow is as follows:

Step 0: Choose a starting solution $\mathbf{m_0}=\mathbf{0}$. Set $\mathbf{r_0}=\mathbf{d}$. Compute $\mathbf{g_0}=\mathbf{G^T d}$. Let $\mathbf{h_0}=\mathbf{g_0}$ be the conjugate direction.

For *k=0* to (*n-1*) do:

- Step 1: $\boldsymbol{\alpha_k} = \frac{\mathbf{g_k^T g_k}}{(\mathbf{Gh_k})^\mathbf{T}\mathbf{Gh_k}}$, and $\mathbf{r_{k+1}} = \mathbf{r_k} - \alpha_k \mathbf{Gh_k}$

- Step 2: $\mathbf{m_{k+1}} = \mathbf{m_k} + \alpha_k \mathbf{h_k}$

- Step 3: $\mathbf{\bar{m}_{k+1}} = \mathbf{F}(\mathbf{m_{k+1}})$

- Step 4: If $\frac{||\mathbf{r_{k+1}} - \mathbf{r_k}||}{||\mathbf{r_{k+1}}||} < \mathbf{0.1}$, then $\mathbf{\bar{m}_{k+1}}$ is the solution, and one quits the loop. Otherwise, continue.

- Step 5: $\mathbf{g_{k+1}} = \mathbf{G^T r_{k+1}}$

- Step 6: $\mathbf{\bar{h}_k} = \frac{\mathbf{\bar{m}_{k+1}} - \mathbf{m_k}}{\boldsymbol{\alpha_k}}$

- Step 7: $\boldsymbol{\beta_k} = \frac{\mathbf{g_{k+1}^T g_{k+1}}}{\mathbf{g_k^T g_k}}$, and $\mathbf{h_{k+1}} = \mathbf{g_{k+1}} + \boldsymbol{\beta}_k \mathbf{\bar{h}_k}$

Appendix D: Modification for Weighted Least-Squares Migration

Seismic data often contain undesirable linear noise, such as head waves. Ideally, such noise should be muted before migrating the data. Similarly, the demigration process may include some data-truncation artifacts due to the nature of inverse Fourier transform used to apply the $i\omega$ operator, causing potentially wrap-around or other artifacts after migration (Figure 2.D-1). Such artifacts in demigrated data must also be muted.

Muting can be understood as a weighting operator, in which the points representing head waves and wrap-around artifacts are set with weights = 0, while the points representing useful data are set with weights = 1. Tapering gives weights between 0 and 1. I need a solid mathematical ground for the weighted least-squares problem.

The objective function that I want to minimize now become

$$J = \|\mathbf{W}(\mathbf{d}-\mathbf{Gm})\|^2 \tag{2.D-1}$$

Expanding J, I have:

$$J = (\mathbf{W}(\mathbf{d}-\mathbf{Gm}))^T\mathbf{W}(\mathbf{d}-\mathbf{Gm}) = (\mathbf{d}-\mathbf{Gm})^T\mathbf{W}^T\mathbf{W}(\mathbf{d}-\mathbf{Gm}) = (\mathbf{d}^T-\mathbf{m}^T\mathbf{G}^T)\mathbf{W}^T\mathbf{W}(\mathbf{d}-\mathbf{Gm})$$

$$= \mathbf{d}^T\mathbf{W}^T\mathbf{Wd} - \mathbf{m}^T\mathbf{G}^T\mathbf{W}^T\mathbf{Wd} - \mathbf{d}^T\mathbf{W}^T\mathbf{WGm} + \mathbf{m}^T\mathbf{G}^T\mathbf{W}^T\mathbf{WGm} \tag{2.D-2}$$

The two middle terms are basically the transpose of each other, and thus are equal to each other, giving

$$J=\mathbf{d}^T\mathbf{W}^T\mathbf{Wd} - 2\mathbf{d}^T\mathbf{W}^T\mathbf{WGm} + \mathbf{m}^T\mathbf{G}^T\mathbf{W}^T\mathbf{WGm} \tag{2.D-3}$$

To minimize J, I need to find where $\nabla J=0$ with respect to $\mathbf{m}$:

$$\nabla J= - 2(\mathbf{d}^T\mathbf{W}^T\mathbf{WG})^T + 2\mathbf{G}^T\mathbf{W}^T\mathbf{WGm} = 0, \text{ or} \tag{2.D-4}$$

$$\mathbf{G}^T\mathbf{W}^T\mathbf{WGm} = \mathbf{G}^T\mathbf{W}^T\mathbf{Wd} \tag{2.D-5}$$

Updating the workflow, I have:

Step 0: Choose a starting solution $\mathbf{m_0}=\mathbf{0}$. Set $\mathbf{r_0}=\mathbf{d}$. Compute $\mathbf{g_0}=\mathbf{G^TW^TWd}$. Let $\mathbf{h_0}=\mathbf{g_0}$ be the conjugate direction.

For *k=1* to *n* do:

- Step 1: $\boldsymbol{\alpha_k} = \frac{\mathbf{g_{k-1}^T g_{k-1}}}{(\mathbf{Gh_{k-1}})^\mathbf{T}\mathbf{W^TWGh_{k-1}}}$, and $\mathbf{r_k}=\mathbf{r_{k-1}} - \alpha_k\mathbf{Gh_{k-1}}$

- Step 2: $\mathbf{m_k}=\mathbf{m_{k-1}}+\alpha_k\mathbf{h_{k-1}}$

- Step 3: $\mathbf{\bar{m}_k} = \mathbf{F}(\mathbf{m_k})$

- Step 4: If $\frac{||\mathbf{r_k}-\mathbf{r_{k-1}}||}{||\mathbf{r_k}||} < \mathbf{0.1}$, then $\mathbf{\bar{m}_k}$ is the solution, and one quits the loop. Otherwise, continue.

- Step 5: $\mathbf{g_k}=\mathbf{G^TW^TWr_k}$

- Step 6: $\mathbf{\bar{h}_k} = \frac{\mathbf{\bar{m}_k}-\mathbf{m_{k-1}}}{\boldsymbol{\alpha_k}}$

- Step 7: $\boldsymbol{\beta_k} = \frac{\mathbf{g_k^T g_k}}{\mathbf{g_{k-1}^T g_{k-1}}}$, and $\mathbf{h_k}=\mathbf{g_k} + \boldsymbol{\beta}_k\mathbf{\bar{h}_k}$

In practice: applying such weights is equivalent to muting the data in the unmigrated domain (i.e. the domain of the original data and the demigrated data). Sometimes, muting is also applied in the migrated data domain, in case the demigrated data exhibit enhanced low-frequency artifacts at far-offset traces (Figure 2.D-2).

Appendix E: Step-by-step Workflow

The complicated, iterative nature of the constrained conjugate gradient least-squares migration makes it difficult to represent the workflow by a simple flow chart. Even the first iteration requires many small steps and generates many different outputs, each to be used in either the same iteration or the next one. Therefore, I choose to represent the workflow with a step-by-step guide for each iteration.

A step-by-step workflow for a maximum of *n* iterations of constrained conjugate gradient least-squares migration is as follows:

Iteration #0 (i.e. preparation iteration):

- Mute the original data to avoid head wave contamination. This is basically multiplying $\mathbf{W}^\mathbf{T}\mathbf{W}$ with data $\mathbf{d}$: $\mathbf{W}^\mathbf{T}\mathbf{W}\mathbf{d}$
- Compute fold and offset map. This step is a part of Kirchhoff migration procedure, but only needs to be done once.
- Migrate the data with anti-aliasing enabled, using the muted original data and the calculated fold and offset map. Anti-aliasing feature in migration reduces migration artifacts. Applying the migration operator $\mathbf{G}^\mathbf{T}$ to $\mathbf{W}^\mathbf{T}\mathbf{W}\mathbf{d}$: $\mathbf{g_0}=\mathbf{G}^\mathbf{T}\mathbf{W}^\mathbf{T}\mathbf{W}\mathbf{d}$
- Apply structure-oriented filtering to $\mathbf{g_0}$: $\mathbf{h_0}=\mathbf{F}(\mathbf{g_0})$. Perform muting if needed. The reason I apply SOF in iteration #0 is to further constrain the result to improve the signal-to-noise ratio. Plus, running SOF in this iteration means I do not need to run SOF for iteration #1, thereby saving one step.
- Demigrate the SOF migrated result. This is equivalent to applying the forward modeling operator $\mathbf{G}$ to $\mathbf{h_0}$: $\mathbf{G}\mathbf{h_0}$
- Mute the demigrated result. Again, multiplying weight $\mathbf{W}^\mathbf{T}\mathbf{W}$ with $\mathbf{G}\mathbf{h_0}$: $\mathbf{W}^\mathbf{T}\mathbf{W}\mathbf{G}\mathbf{h_0}$

Iteration #1:

- Update the residual: calculate α_1 and $\mathbf{r_1}$:

$\alpha_1 = \frac{\mathbf{g_0^T g_0}}{(\mathbf{Gh_0})^\mathbf{T}\mathbf{W^T W G h_0}}$ and $\mathbf{W^T W r_1} = \mathbf{W^T W d} - \alpha_1 \mathbf{W^T W G h_0}$

The denominator of α_1 is basically the square of muted demigrated result $\mathbf{W^T W G h_0}$.

Since $\mathbf{W}$ consists of 0 and 1 only (i.e. muted: w=0, non-muted: w=1), $\mathbf{W^T W} = \mathbf{W}$, and thus

$(\mathbf{W^T W G h_0})^\mathbf{T}(\mathbf{W^T W G h_0}) = (\mathbf{W G h_0})^\mathbf{T}(\mathbf{W G h_0}) = (\mathbf{G h_0})^\mathbf{T}\mathbf{W^T W}(\mathbf{G h_0})$

Note that in the residual calculation, I use the muted result of original data and demigrated data, and thus the updated residual is muted and I can skip the muting step later on.

- Update the model: $\mathbf{m_1}=\mathbf{m_0}+\alpha_1\mathbf{h_0}=\mathbf{0}+\alpha_1\mathbf{h_0}=\alpha_1\mathbf{h_0}$ (because $\mathbf{m_0}$ is initialized to $\mathbf{0}$). Since $\mathbf{h_0}$ is the SOF-applied result, I can skip the constrain step.
- Migrate the muted updated residual with anti-alias disabled (since I do not want the residual to be too smoothed): $\mathbf{g_1}=\mathbf{G^T W^T W r_1}$
- (Optional) Mute the migrated residual if needed.
- Update the conjugate gradient: calculate β_1 and $\mathbf{h_1}$:

$\beta_1 = \frac{\mathbf{g_1^T g_1}}{\mathbf{g_0^T g_0}}$, and $\mathbf{h_1}=\mathbf{g_1} + \beta_1\mathbf{h_0}$

Note that I skip the step of calculating $\bar{\mathbf{h}}_\mathbf{1}$. This is because:

$$\bar{\mathbf{h}}_\mathbf{1}=\frac{\bar{\mathbf{m}}_\mathbf{1}-\mathbf{m_0}}{\alpha_1}=\frac{\bar{\mathbf{m}}_\mathbf{1}}{\alpha_1}=\frac{\alpha_1 * \mathbf{h_0}}{\alpha_1}=\mathbf{h_0}$$

- Demigrate the updated conjugate gradient: $\mathbf{Gh_1}$
- Mute the demigrated result: $\mathbf{W^T W G h_1}$

Iteration #k ($2 \le k < n$):

- Update the residual: calculate α_k and $\mathbf{r_k}$:

$\alpha_k = \frac{\mathbf{g_{k-1}^T g_{k-1}}}{\mathbf{(Gh_{k-1})^T W^T W G h_{k-1}}}$ and $\mathbf{W^T W r_k} = \mathbf{W^T W r_{k-1}} - \alpha_k \mathbf{W^T W G h_{k-1}}$

- Update the model: $\mathbf{m_k} = \mathbf{m_{k-1}} + \alpha_k \mathbf{h_{k-1}}$
- Apply structure-oriented filtering on $\mathbf{m_k}$: $\bar{\mathbf{m}}_\mathbf{k} = \mathbf{F}(\mathbf{m_k})$
- Update the directional vector: $\bar{\mathbf{h}}_\mathbf{k} = \frac{\bar{\mathbf{m}}_\mathbf{k} - \mathbf{m_{k-1}}}{\alpha_k}$
- Migrate the muted updated residual with anti-alias disabled: $\mathbf{g_k} = \mathbf{G^T W^T W r_k}$
- (Optional) Mute the migrated residual if needed.
- Update the conjugate gradient: calculate β_k and $\mathbf{h_k}$:

$\beta_k = \frac{\mathbf{g_k^T g_k}}{\mathbf{g_{k-1}^T g_{k-1}}}$, and $\mathbf{h_k} = \mathbf{g_k} + \beta_k \bar{\mathbf{h}}_\mathbf{k}$

- Demigrate the updated conjugate gradient: $\mathbf{G h_k}$
- Mute the demigrated result: $\mathbf{W^T W G h_k}$

Iteration #n (last iteration):

- Update the residual: calculate α_n and $\mathbf{r_n}$:

$\alpha_n = \frac{\mathbf{g_{n-1}^T g_{n-1}}}{\mathbf{(Gh_{n-1})^T W^T W G h_{n-1}}}$ and $\mathbf{W^T W r_n} = \mathbf{W^T W r_{n-1}} - \alpha_n \mathbf{W^T W G h_{n-1}}$

- Update the model: $\mathbf{m_n} = \mathbf{m_{n-1}} + \alpha_n \mathbf{h_{n-1}}$
- Apply structure-oriented filtering on $\mathbf{m_n}$: $\bar{\mathbf{m}}_\mathbf{n} = \mathbf{F}(\mathbf{m_n})$

References

Biondi, B., 2001, Kirchhoff imaging beyond aliasing: Geophysics, **66(2)**, 654-666.

Dev, A. and G. A. McMechan, 2009, Spatial antialias filtering in the slowness-frequency domain, Geophysics, **74(2)**, V35-V42.

Guo, S., B. Zhang, Q. Wang, A. Cabrales-Vargas, and K. J. Marfurt, 2016, Noise suppression using preconditioned least-squares prestack time migration: Application to the Mississippian Limestone: Journal of Geophysics and Engineering, **13**, 441-453.

Ha, T. and K. J. Marfurt, 2017, Seismic reprocessing and interpretation of a fractured-basement play: Texas Panhandle. Interpretation, manuscript submitted for publication.

Hestenes, M. and E. Stiefel, 1952, Methods of Conjugate Gradient for Solving Linear Systems: Journal of Research of the National Bureau of Standards, **49(6)**, 409-436.

Lewis, J. M., S. Lakshmivarahan, and S. K.Dhall, 2006, Dynamic Data Assimilation: a Least Squares Approach. Cambridge: Cambridge University Press.

Nemeth, T., C. Wu, and G. T. Schuster, 1999, Least-squares migration of incomplete reflection data: Geophysics, **64(1)**, 208-221.

Nemeth, T., H. Sun, and G. T. Schuster, 2000, Separation of signal and coherent noise by migration filtering: Geophysics, **65(2)**, 574-583.

Pramik, B., 2011, Broadband land acquisition – Survey design issues, SEG Technical Program, Expanded Abstract 2011, 4344-4348.

Sorenson, R., 2005, A dynamic model for the Permian Panhandle and Hugoton fields, western Anadarko basin: AAPG Bulletin, **89**, 921-938.

Verma, S., S. Guo, T. Ha, and K. J. Marfurt, 2016, Highly aliased ground-roll suppression using a 3D multiwindow Karhunen-Loeve filter: Application to a legacy Mississippi Lime survey: Geophysics, **81**, V79-V88.

Yu, J., J. Hu, G. T. Schuster, and R. Estill, 2006, Prestack migration deconvolution: Geophysics, **71(2)**, S53-S62

Zeng, C., S. Dong, and B. Wang, 2014, Least-squares reverse time migration: Inversion-based imaging toward true reflectivity: The Leading Edge, **33(9)**, 962–968.

Zeng, C., S. Dong, and B. Wang, 2017, A guide to least-squares RTM for subsalt imaging: challenges and solutions: Interpretation, Manuscript accepted for publication.

Zhang, B., T. Lin, S. Guo, O. E. Davogustto, and K. J. Marfurt, 2016, Noise suppression of time-migrated gathers using prestack structure-oriented filtering: Interpretation, **4**, SG19-SG29.

Figures

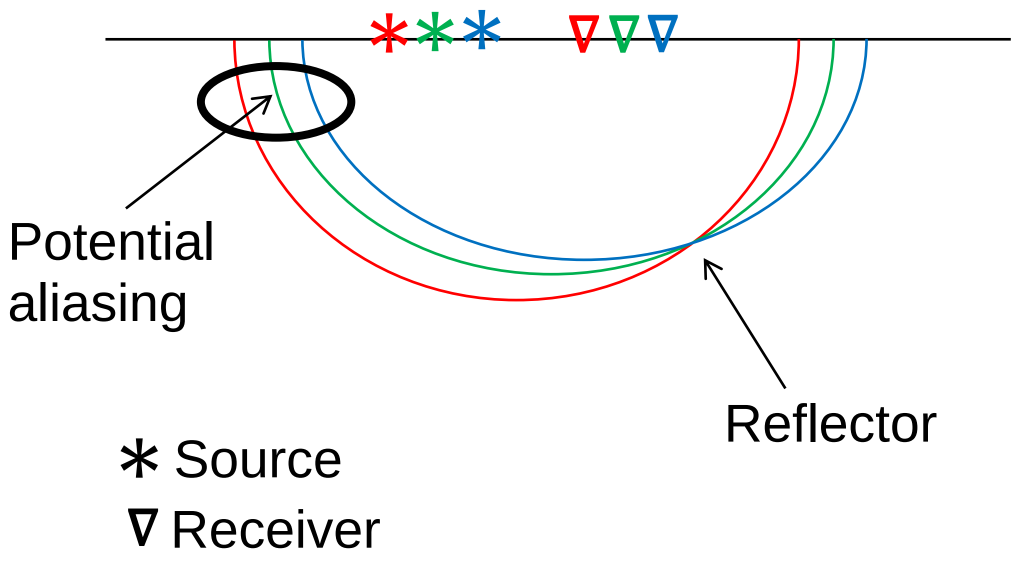


Figure 2.1. Schematic showing the migration process. The main idea is to copy each amplitude value along ellipsoids and then stack all the subsequent images together. If the data are sufficiently sampled, some areas constructively interfere into reflectors, while others destructively interfere into zero-data zones. If the data are insufficiently sampled, there may be only partial destructive interference, resulting in aliasing.

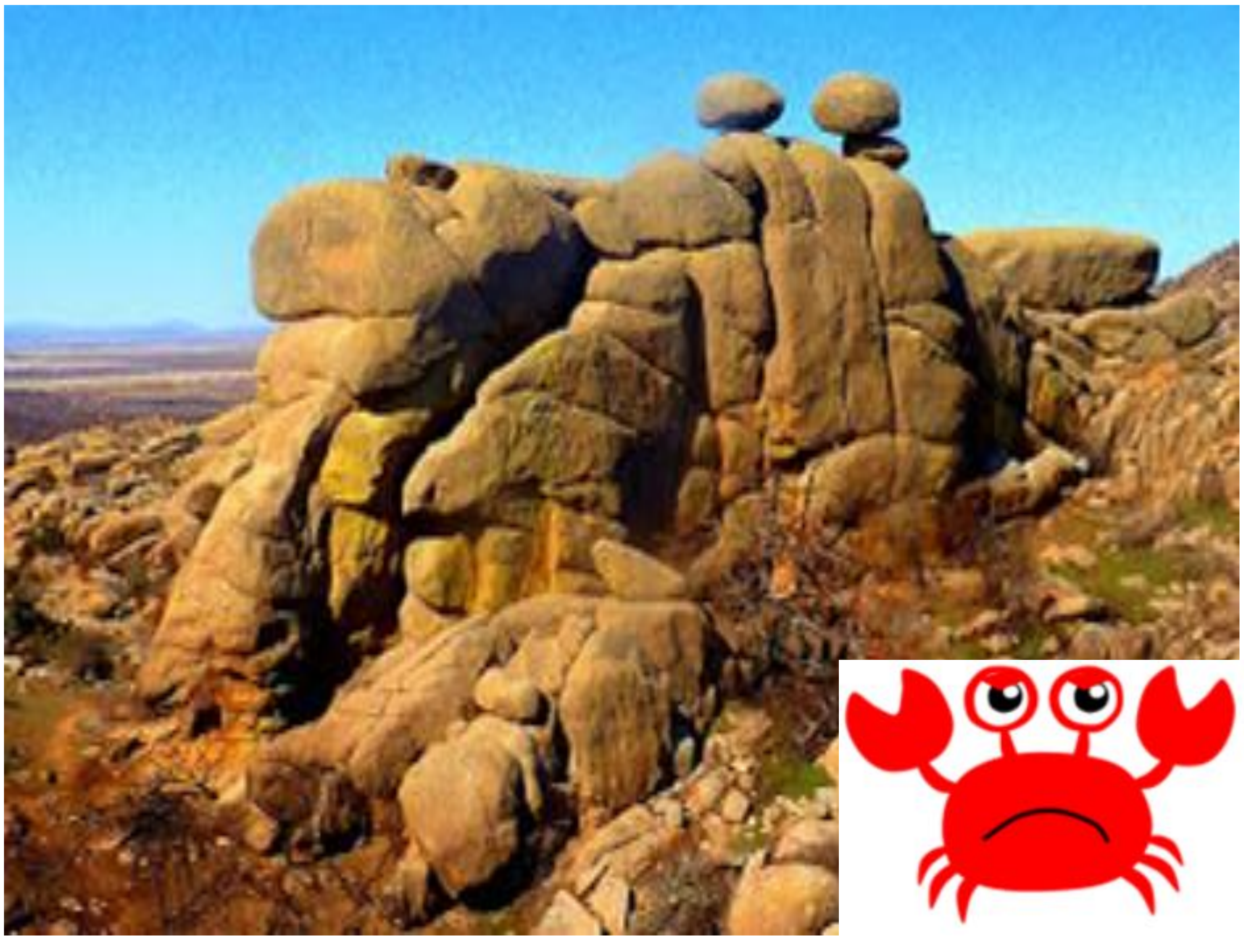


Figure 2.2. Crab-eye rock at Charon’s Garden, Wichita Mountains (see how it looks like a frowny crab?). The rock is composed of fractured granite with multiple sets of joints that are several tens of feet apart. This is an outcrop analog to the fractured basement 2500-ft below the ground.

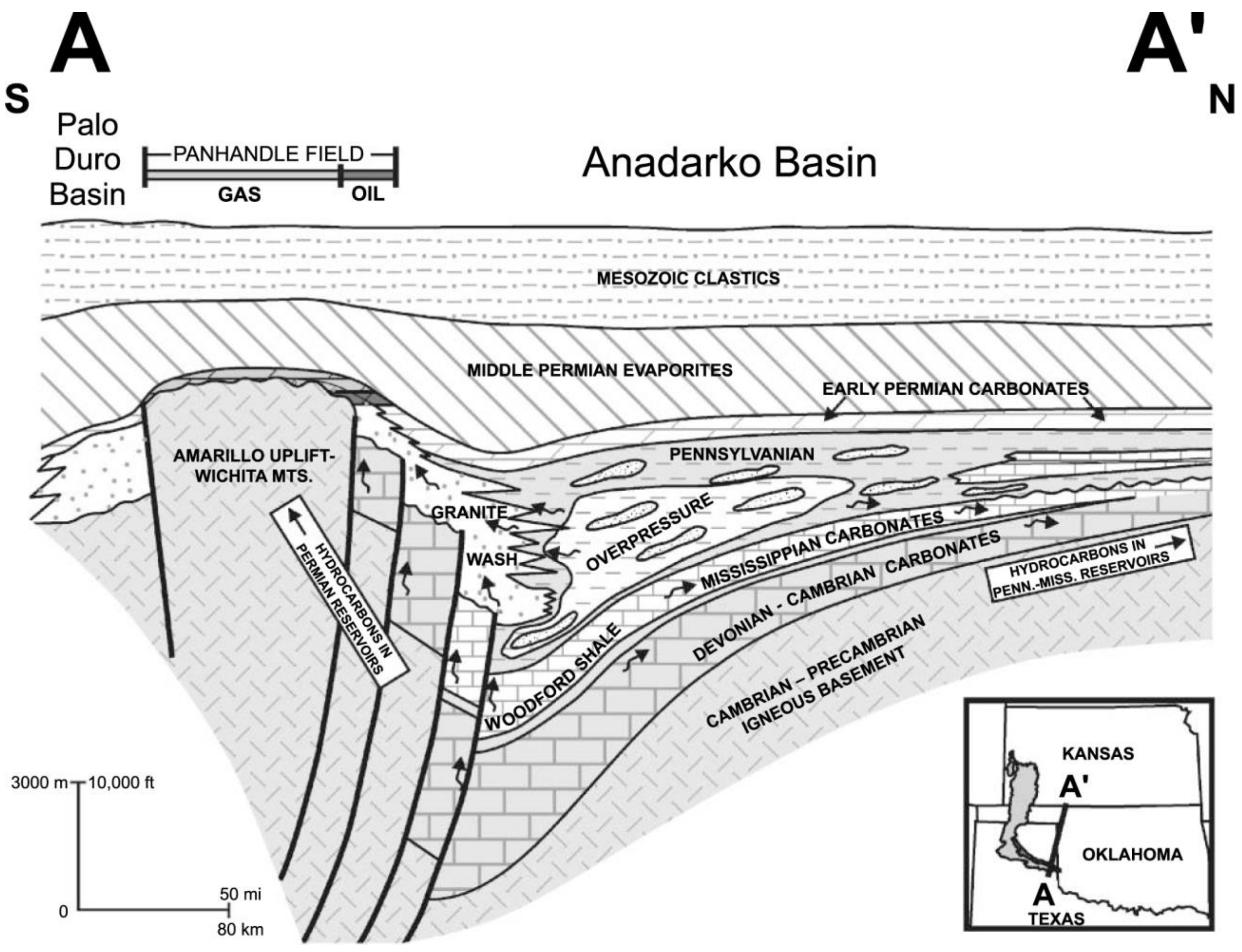


Figure 2.3. Regional geological cross-section through the Panhandle field (Sorenson, 2005). Source rocks are located in the deeper part of the Anadarko Basin and have an age range from Ordovician to Pennsylvanian, including the Mississippian Woodford Shale. The most common reservoir rocks are the early Permian carbonate and the Granite Wash. Oil also fills joints and fractures that formed in the previously exposed basement highs. Above the reservoir rocks, middle Permian evaporites act as a seal. Such a thick, high-velocity layer of evaporite is the cause of strong head waves in seismic data.

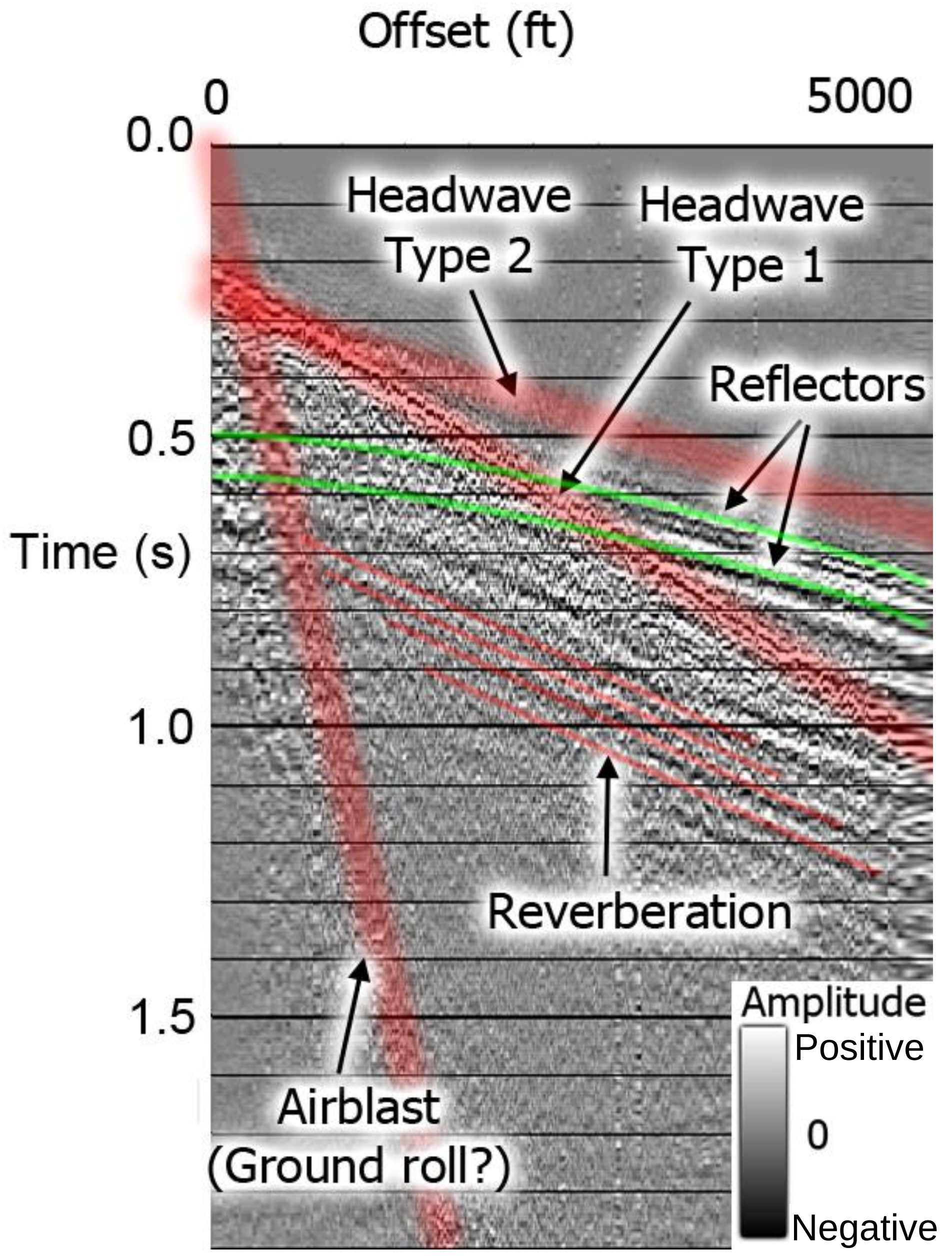


Figure 2.4. A representative 3D shot gather sorted by offset with interpreted events, including head waves, reflections, air blast (or ground roll?), and reverberations. At the target top basement depth (t=0.57s), critical refraction occurs at offset h = 3200ft. Beyond this point, the signal are highly contaminated by coherent, moderate bandwidth head waves.

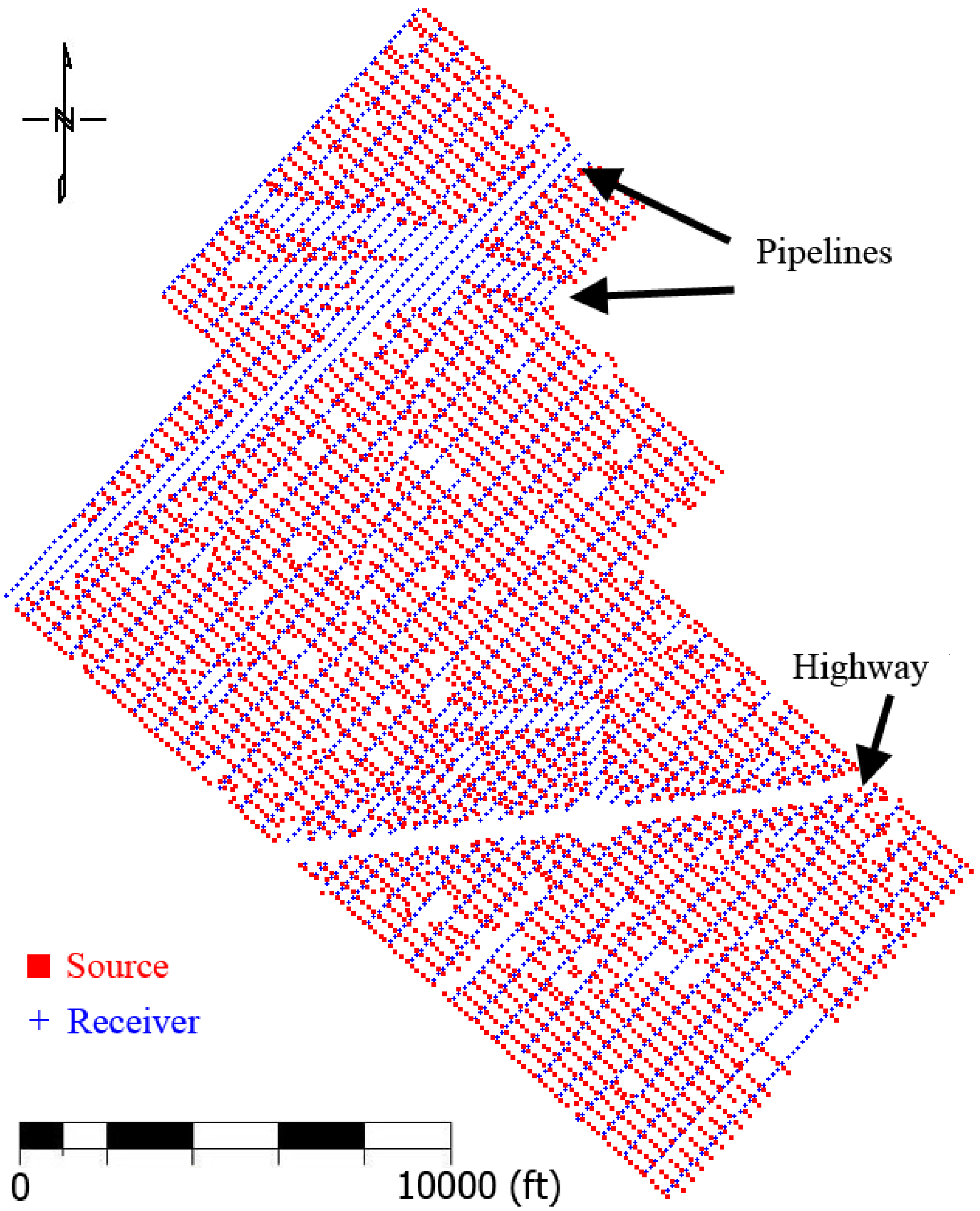


Figure 2.5. Source (red squares) and receiver (blue crosses) geometry of the seismic survey. Linear gaps in source and receiver locations are associated with roads. Other smaller, circular gaps are areas inaccessible to vibroseis trucks. These gaps, together with the rectangular gridding geometry, generate acquisition footprint in seismic data, especially at shallow target depth.

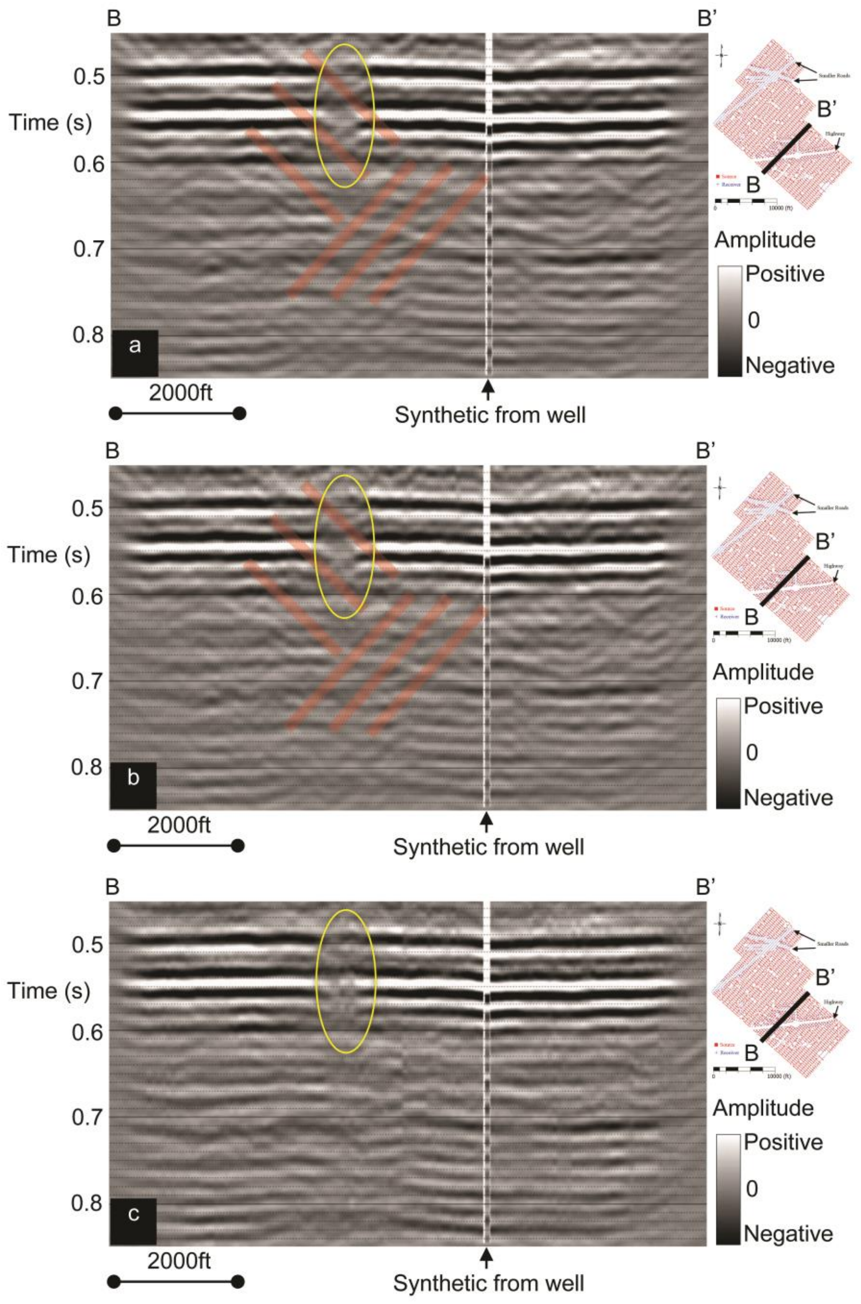


Figure 2.6. Vertical slice through the seismic amplitude volume generated from (a) conventional Kirchhoff prestack time migration, (b) unconstrained conjugate gradient least-squares prestack time migration, and (c) constrained conjugate gradient least-squares prestack time migration. The same migration algorithm is used in all cases. Note the cross-cutting migration artifacts (red lines) in the conventional migrated image that are enhanced in the least-squares migrated image without the constraint. Also note a sudden amplitude decrease (yellow ellipses) where a highway intersects the profile. The constrained conjugate gradient least-squares migration suppresses those artifacts and provides a better amplitude balancing.

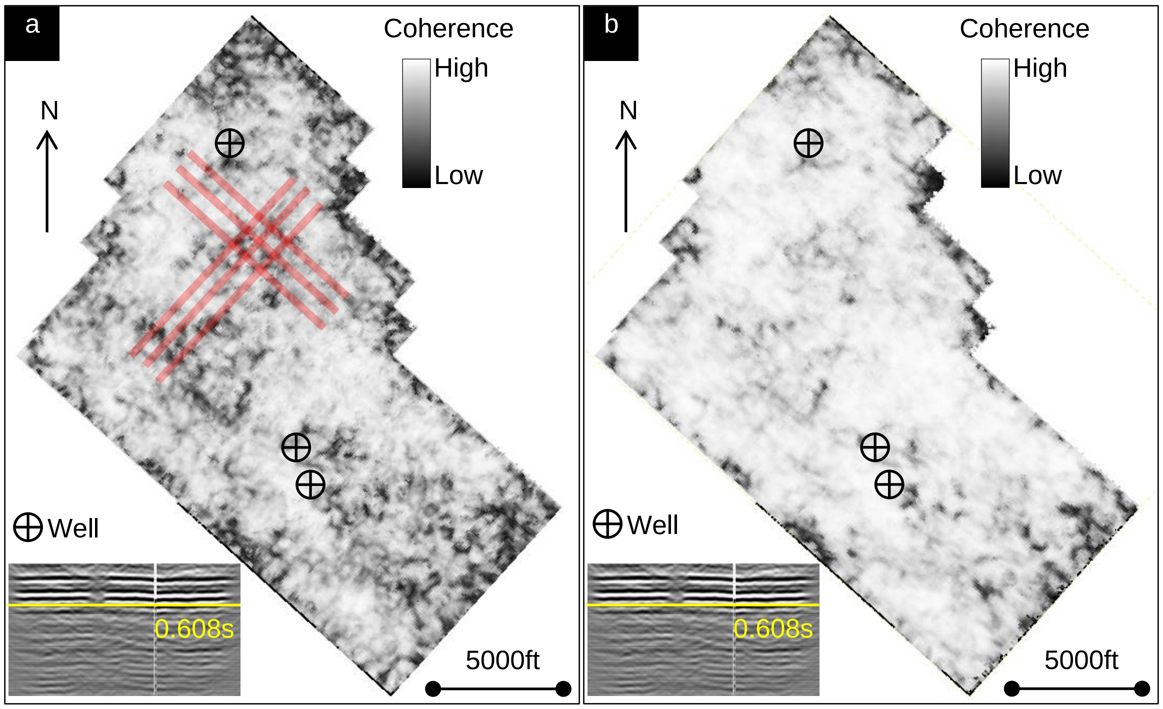


Figure 2.7. Coherence time slice at t=0.608s (close to top basement) generated from (a) conventional Kirchhoff prestack time migration and (b) constrained conjugate gradient least-squares prestack time migration. Most of the grid-like low-coherence hash pattern (red lines) seen in the conventional migrated coherence map is suppressed in the constrained conjugate gradient least-squares migrated coherence map.

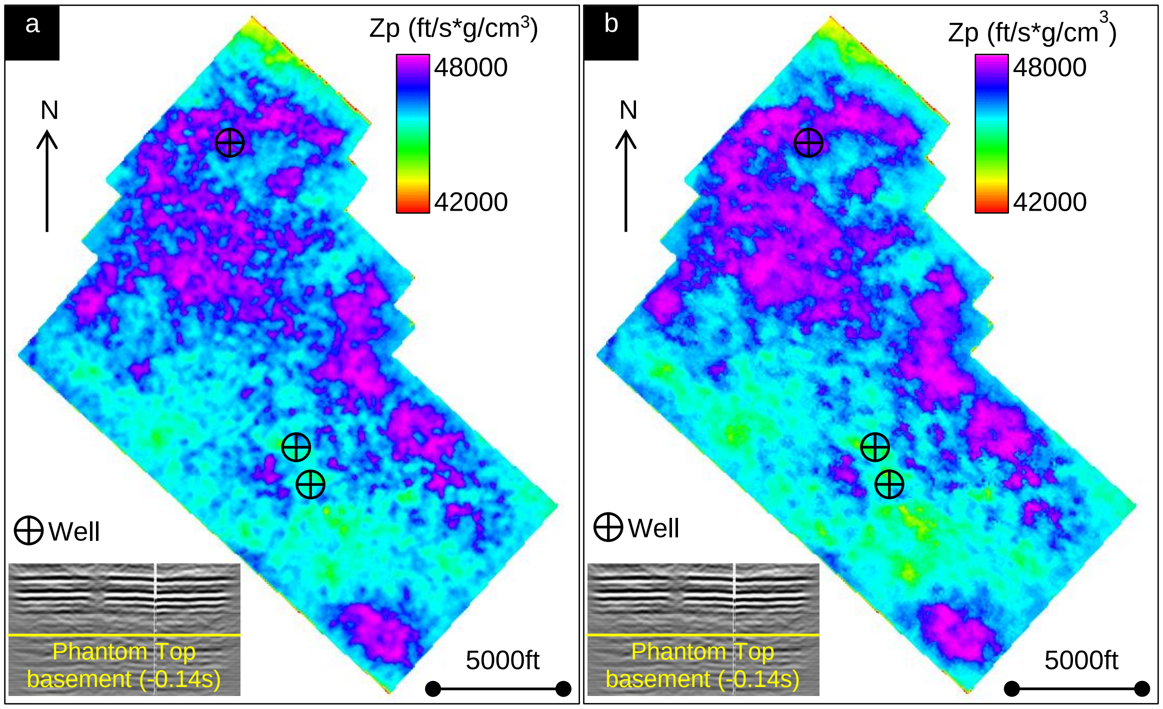


Figure 2.8. Phantom horizon map 0.14 s below the top basement through P-impedance volumes generated from (a) conventional Kirchhoff prestack time migration and (b) constrained conjugate gradient least-squares prestack time migration. Impedance map created by constrained conjugate gradient least-squares migration exhibits less hash-pattern noise, making it easier to isolate zones of low impedance corresponding to potential open fractures filled with hydrocarbons.

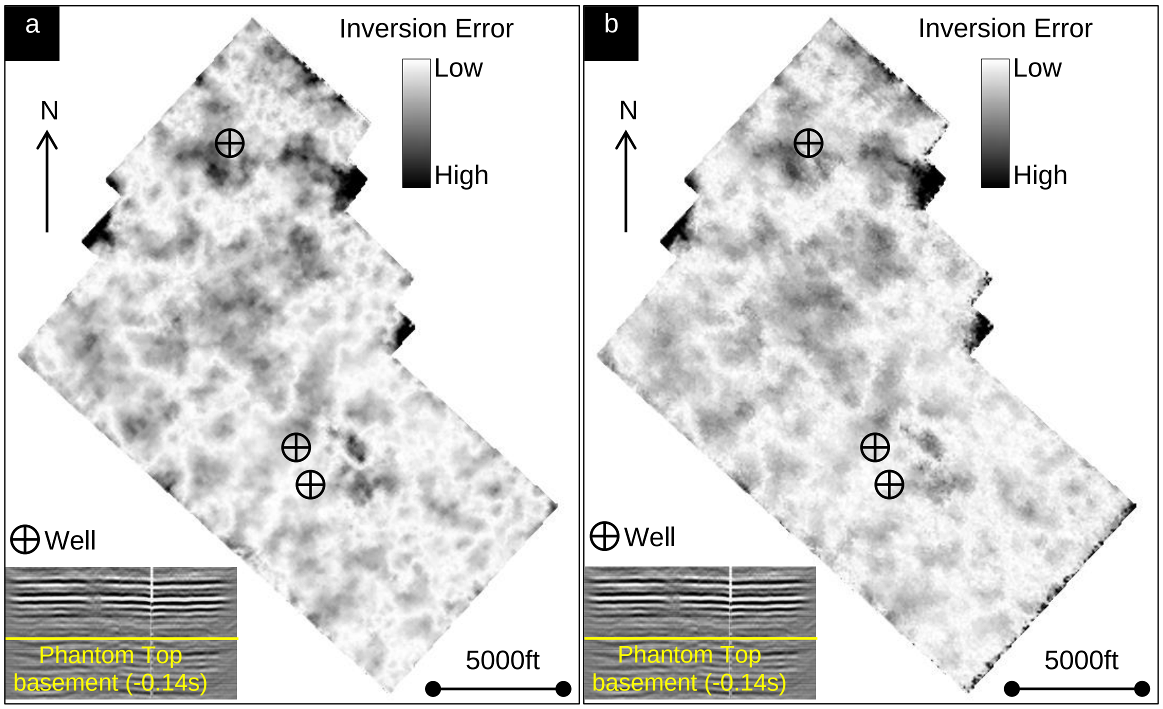


Figure 2.9. Phantom horizon map 0.14s below the top basement through inversion misfit error volume generated from (a) conventional Kirchhoff prestack time migration and (b) constrained conjugate gradient least-squares prestack time migration. The constrained conjugate gradient least-squares migration errors are smoother and contains less hash-pattern artifacts than those created by conventional Kirchhoff migration.

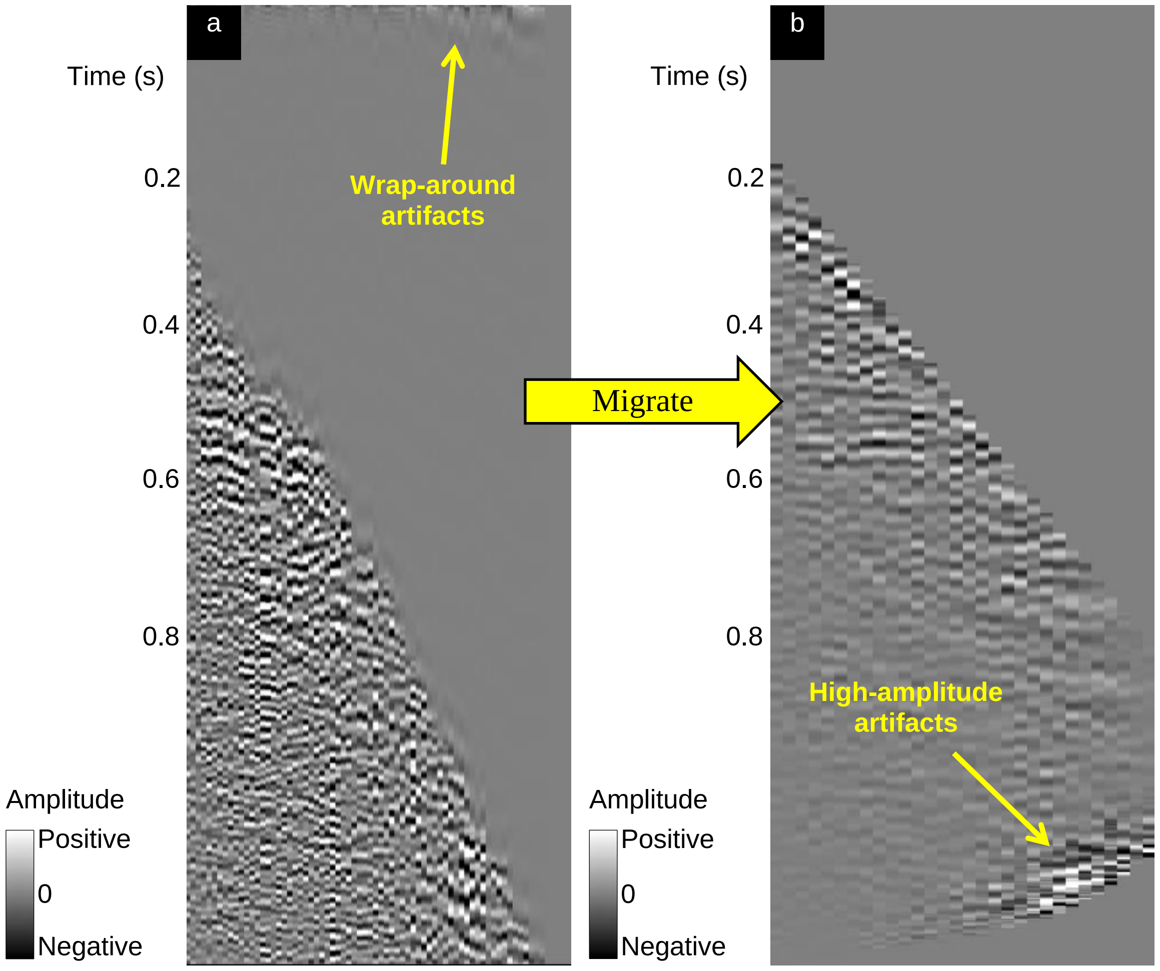


Figure 2.D-1. (a) A demigrated CDP gather showing wrap-around artifacts close to time zero. These artifacts are inherent to the forward modeling (i.e. demigration) operator, due to the nature of the inverse Fourier transform. (b) Migrated result of demigrated data in (a). The high-amplitude artifacts at the bottom of the migrated CDP gather are caused by such wrap-around artifacts. Thus, it is important to mute (or alternatively, sufficiently pad) the top part of the demigrated gathers.

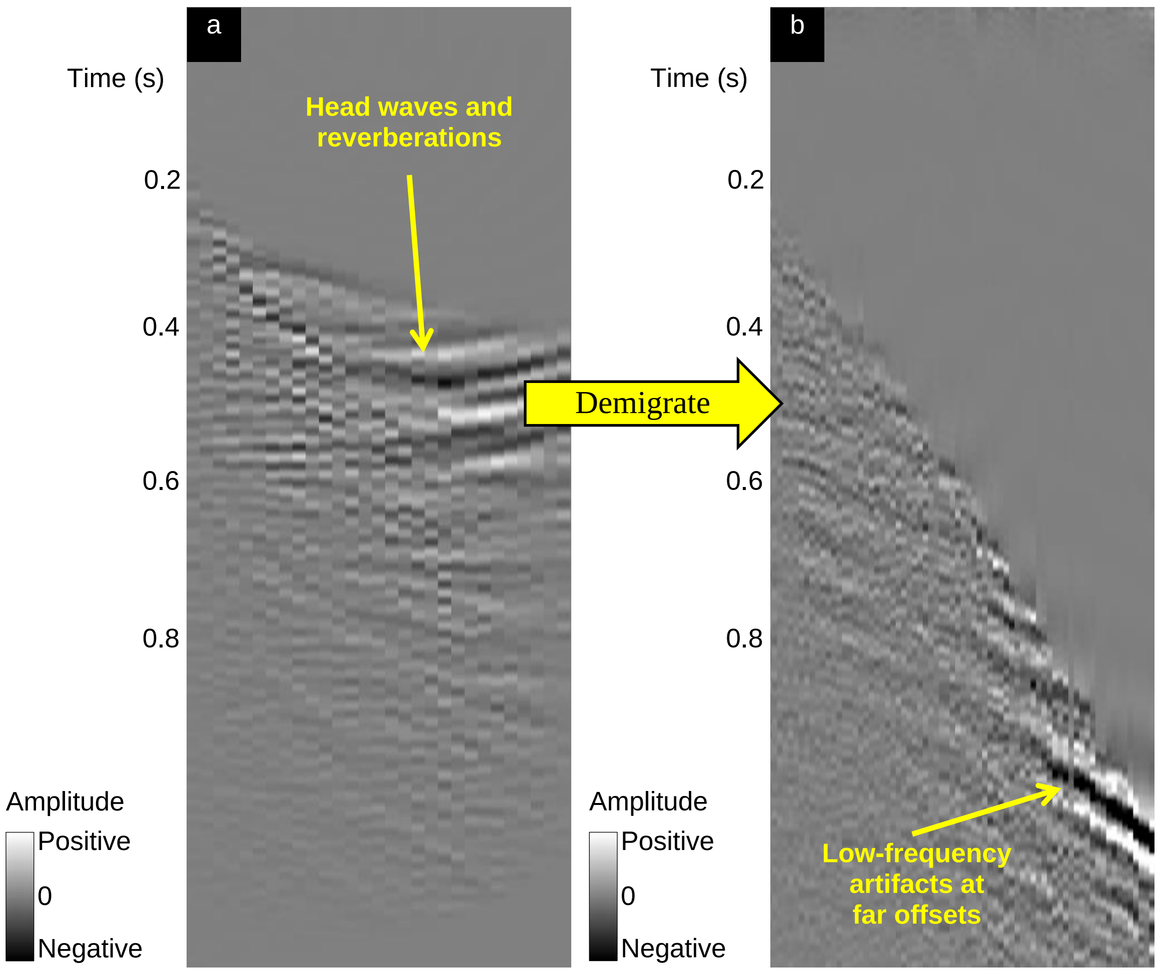


Figure 2.D-2. (a) A migrated CDP gather showing head waves and reverberations. (b) Demigrated result of the migrated data in (a). The low-frequency artifacts at far offset in demigrated gather are caused by inadequate suppression of head waves and reverberations. Thus, if the muting and noise suppression applied on the original data is insufficient to remove head waves and reverberations, the migrated result must be muted as well.

CHAPTER 3: PITFALLS AND IMPLEMENTATION OF DATA CONDITIONING, ATTRIBUTE ANALYSIS, AND SELF-ORGANIZING MAP TO 2D DATA: APPLICATION TO THE NORTH CARNARVON BASIN, AUSTRALIA

Abstract

Recent developments in attribute analysis and machine learning have significantly enhanced interpretation workflows of 3D seismic surveys. Nevertheless, even in 2018, many sedimentary basins are only covered by grids of 2D seismic lines. These 2D surveys are suitable for regional feature mapping and often identify targets in areas not covered by 3D surveys. With continuing pressure to cut costs in the hydrocarbon industry, it is crucial to extract as much information as possible from these 2D surveys. Unfortunately, much if not most modern interpretation software packages are designed to work exclusively with 3D data. To determine if I can apply 3D volumetric interpretation workflows to grids of 2D seismic lines, I perform data conditioning, attribute analysis, and a machine learning technique called self-organizing map (SOM) to the 2D data acquired over the Exmouth Plateau, North Carnarvon Basin, Australia. I find that these workflows allow me to significantly improve image quality, interpret regional geological features, identify local anomalies, and perform seismic facies analysis. However, these workflows are not without pitfalls. I need to be careful in choosing the order of filters in data conditioning workflow and be aware of reflector misties at line intersections. Vector data, such as reflector convergence, need to be extracted and then mapped component-by-component before combining the results. I am also unable to perform attribute extraction along a surface or geobody extraction for 2D data in a commercial interpretation software package. To address this issue, I devise a point-by-point attribute extraction workaround to overcome the incompatibility between 3D interpretation workflow and 2D data.

Introduction

Before attempting to interpret seismic attributes computed from 2D surveys, it is crucial to understand the differences between 2D and 3D data. Hutchinson (2016) provides a detailed discussion regarding the limitations and advantages of 2D versus 3D data in five aspects: (1) processing artifacts, (2) sharpness of discontinuities, (3) reflector dip, (4) faults, and (5) amplitude contrast. First, due to imaging of out-of-the-plane energy into the vertical plane defined by the 2D seismic line in the migration process, 2D data usually show cross-cutting and bow-tie artifacts that are otherwise properly focused in 3D data (Figure 3.1). This imaging limitation of 2D data also results in blurred discontinuities (Figure 3.2) and incorrect reflector dip (Figure 3.3) compared to 3D data. In general, a skilled interpreter can pick similar fault "sticks" on 2D data as with 3D data, although the fault connectivity between lines may differ (Figure 3.4). Because of acquisition constrains, the shallow amplitude contrast is often better on 2D data than on 3D data (Figure 3.5).

It is also important to consider the limitations of each attribute type with respect to the number of data dimensions (Table 3.1). While all 3D attributes appear to be compatible with 2D data, the majority of them assume a 2.5D earth model, in which the earth looks the same in the direction perpendicular to the 2D line. In other words, there is no azimuthal information for 2D attribute calculation (e.g. there are only *apparent* dip and *apparent* reflector convergence).

A conventional 2D interpretation workflow starts with picking a horizon – a geological boundary of interest. This horizon is then interpolated to form a surface, where the two-way-travel time of this surface constitute a time-structure map of the geological boundary. The next step is to compute surface attributes, such as dip magnitude, dip azimuth, and possibly curvature of this surface to further aid the interpretation of the structural relief of the geological boundary.

Note that these surface attributes are calculated on a surface rather than on the seismic data itself. Thus, the quality of these surface attributes highly depends on the interpreter. To overcome this dependency, Bahorich and Bridges (1992) created the Seismic Sequence Attribute Mapping (SSAM) workflow, which extracts seismic attributes computed from the 2D amplitude data and maps them as a surface. The few published SSAM workflows focused primarily on single-trace attributes (e.g. instantaneous envelope, frequency, and phase) rather than multi-trace attributes (e.g. inline dip, coherence, and GLCM attributes). Another challenge of this workflow is that 2D seismic attribute extraction is not available in many modern software packages that are designed to efficiently handle 3D data volumes.

I begin this chapter with a brief review of the geologic settings of the North Carnarvon Basin. I then describe my 2D seismic interpretation workflow, including data conditioning, attribute analysis, and SOM facies classification. Next, I discuss my interpretation of regional geology and local anomalies from 2D seismic amplitude and attribute profiles. Finally, I show the pitfalls encountered during my interpretation workflow, as well as workarounds to avoid those pitfalls and to correctly display horizons and geobodies extracted from the 2D seismic attribute profiles.

Geologic Settings

The North Carnarvon Basin is a major hydrocarbon reserve in Australia (Chongzhi et al., 2013) that can be divided into several sub-basins (Figure 3.6). Among these sub-basins, the Exmouth Plateau is the largest and contains most of the major gas fields. Thanks to these gas fields, numerous seismic surveys have been acquired over the area. The 3D surveys are relatively small (<5000 km^2), concentrated around known reservoir locations. In contrast, the 2D surveys are much larger, allowing the interpretation of regional features and the identification of new area that has not been covered by 3D data.

The North Carnarvon Basin was a passive margin that underwent multiple stages of extension, subsidence, and late minor inversion, resulting in NE-SW trending faults (Mihut and Muller, 1998; Magee et al.,2013; McArdle et al., 2013; Tellez Rodriguez, 2015). The depositional history of the North Carnarvon Basin can be summarized by the simplified stratigraphic column shown in Table 3.2. The main source rocks of the Exmouth Plateau are the Locker Shale, deposited in early Triassic. The Mungaroo (middle-late Triassic) fluvio-deltaic formation and the early Cretaceous Flag sandstone of the Barrow delta are the main reservoir rocks. During the middle Cretaceous, post-rifting subsidence enabled a thick deposition of the transgressive Muderong Shale throughout the entire basin, which act as a regional seal (Chongzhi et al., 2013). During the Tertiary, the Australian Plate drifted northward to warmer tropical zones, facilitating development of carbonate sequences, including the Mandu Limestone formation (Smith, 2014).

My project involves 55 lines forming a rectangular grid, spanning an area of ~40,000 km^2 in the center of the Exmouth Plateau (Figure 3.6). These 2D lines were acquired five to ten years before the 3D surveys, suffering from low image quality and inaccurate imaging of out-of-the-plane energy due to the limitations in acquisition and processing. My goal is to improve seismic interpretation of these 2D lines in order to have a better understanding of the regional geology as well as possible local hydrocarbon accumulations.

Method

My interpretation workflow consists of data conditioning, seismic attribute analysis, followed by self-organizing map (SOM) classification.

Data conditioning

To improve seismic image quality, I implement a data conditioning workflow developed by Hutchinson et al. (2016) by applying spectral balancing and edge-preserving structure-oriented filtering (Marfurt, 2006) to the data. In my data conditioning workflow, I need to set the analysis window's half-width to five times the bin size of the 2D line due to the high amount of noise and cross-cutting artifacts. Appendix A shows a detail list of data conditioning parameters.

Figure 3.7 compares a shallow section of a seismic line before and after data conditioning. Thin layers are better resolved (yellow ellipse), cross-cutting migration artifacts are suppressed (cyan ellipse), while faults are sharper (red arrows), providing an improved interpretation. The same benefits can be observed in a deeper section of the same seismic line (Figure 3.8).

Seismic attribute analysis

In order to choose which attributes to be used in my analysis, I examine the attribute expression of the seismic facies that I want to identify (Table 3.3). I choose conformal reflectors, anomalously strong reflectors, Mass-Transport Complexes (MTCs), and channels as four major seismic facies in my project. Based on previous works by Qi et al. (2016), Wallet (2016), and Zhao et al. (2016) on channel and MTC interpretation, I choose eight seismic attributes:

- seismic amplitude: not used in seismic facies analysis but for traditional horizon picking and structural interpretation,
- energy-ratio-similarity: a high-resolution coherence attribute to differentiate faults, stratigraphic edges, as well as rotated blocks and chaotic features internal to MTCs from more continuous reflector events (Lin et. al., 2016),
- structural curvature: a measure of folding and flexing, used to differentiate channel axes, levees, and the toe thrust component of MTCs from planar reflectors (Mai et. al., 2009),
- reflector convergence: measures vertical changes in the dip vector, used to differentiate conformal reflectors from pinch-outs, angular unconformities, and chaotic reflector orientation (Marfurt and Rich, 2010),
- coherent energy: the energy of a window of amplitude data, used to differentiate strong coherent from weak incoherent reflectors (Chopra and Marfurt, 2007),
- Gray-Level Co-occurrence Matrix (GLCM) energy and homogeneity: texture attributes that measures the variation in lateral seismic response, used to differentiate constant amplitude and smoothly varying from laterally chaotic reflectors as well as from noise (Matos et. al., 2011), and
- peak frequency: the mode of the decomposed spectral components that is used to differentiate thick (lower peak frequency) from thin (higher peak frequency) layers (Marfurt, 2018).

Details of attribute calculation parameters are shown in Appendix B. Different seismic facies have different distinctive expressions. For example, the slump component of an MTC exhibits overall low coherence, while the related block component and conformal reflectors exhibit high coherence. Channels are usually coherent near their axes, but incoherent at their edges. Among the four facies examined in our data, incised channels express the highest curvature and reflector convergence with the highest spatial variation.

Due to the limitation of different attributes regarding data dimensionality (Table 3.1), I have to make various modifications to the filters and attribute calculators in order to perform 3D attribute analysis on 2D data. Among all data conditioning tasks and selected attributes for my analysis, only spectral balancing/decomposition works out-of-the-box for 2D data, thanks to its trace-by-trace computation nature. For all other attributes, I have to set the crossline analysis window size to zero and disable any azimuth or strike outputs, since 2D data do not contain any azimuthal information. Some attributes, such as dip, curvature, and reflector convergence, have different mathematical formulas between 2D and 3D computations, thus requiring me to program each case in a separate code section.

Since my data consists of 55 individual 2D seismic lines, computing attributes for one line at a time is a tedious task. In general, each line forms a separate file. In order to perform attribute calculation on all of the 2D lines at once, I merge those lines into a pseudo-3D volume, in which the length of the inline axis is the length of the longest line, and the length of the crossline axis is the total number of lines (Figure 3.9). Shorter lines are padded with dead traces until they reach the length of the longest line. Then, this new pseudo-3D volume is input into 3D attribute calculators, with the size of the crossline analysis window set to zero. After the computation, I delete the dead traces and reassign each “inline” back to the appropriate line in the 2D survey.

Self-organizing maps

One way to combine multiple attributes for facies analysis is to use self-organizing map (SOM), an automatic, unsupervised machine learning technique. This process takes *N* attributes residing in an *N*-dimensional space and projects them onto a deformed 2D manifold. These projected data are then mapped against a 2D color table in such a way that voxels within a cluster on the 2D latent space have similar colors. A detailed description of SOM is provided by Zhao et al. (2016).

Qi et al. (2016) finds the choice of attributes to be critical to both interactive interpretation and machine learning algorithms like SOM. Examining Table 3.3, I note that MTCs are characterized by low coherence, low-to-moderate coherent energy, low GLCM energy, and low GLCM homogeneity. During my experiment, I find that continuous reflectors can have the same expression in reflector convergence, structural curvature, and peak frequency as MTCs if those reflectors are slightly undulating and located around the same depth as the MTCs. Therefore, reflector convergence, structural curvature, and peak frequency are not effective in differentiating MTCs from the continuous reflector background in SOM classification. I do not include the seismic amplitude in the SOM analysis either. Rather, I choose to co-render SOM result with seismic amplitude after the SOM computation. Thus, my list of inputs for SOM classification consists of four attributes: energy-ratio-similarity, coherent energy, GLCM energy, and GLCM homogeneity.

To suppress “salt-and-pepper” effects in unsupervised classification algorithm, I follow an attribute preconditioning workflow developed by Qi et. al. (2016). I first smooth the attributes along structural dip for two iterations, then apply a Kuwahara median filter to the input attributes. Appendix C shows a flowchart of attribute preconditioning and detail parameters of my SOM workflow.

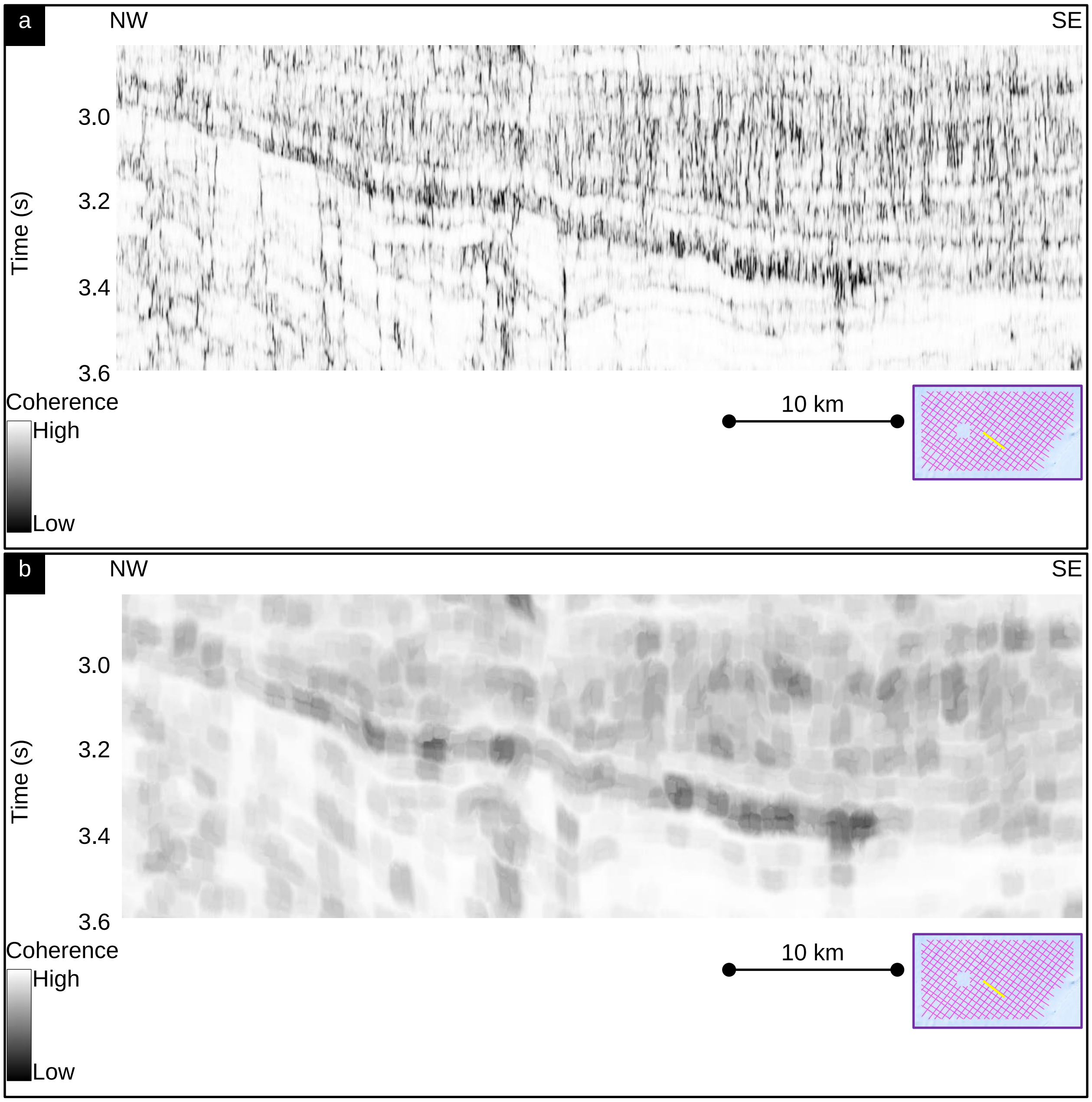
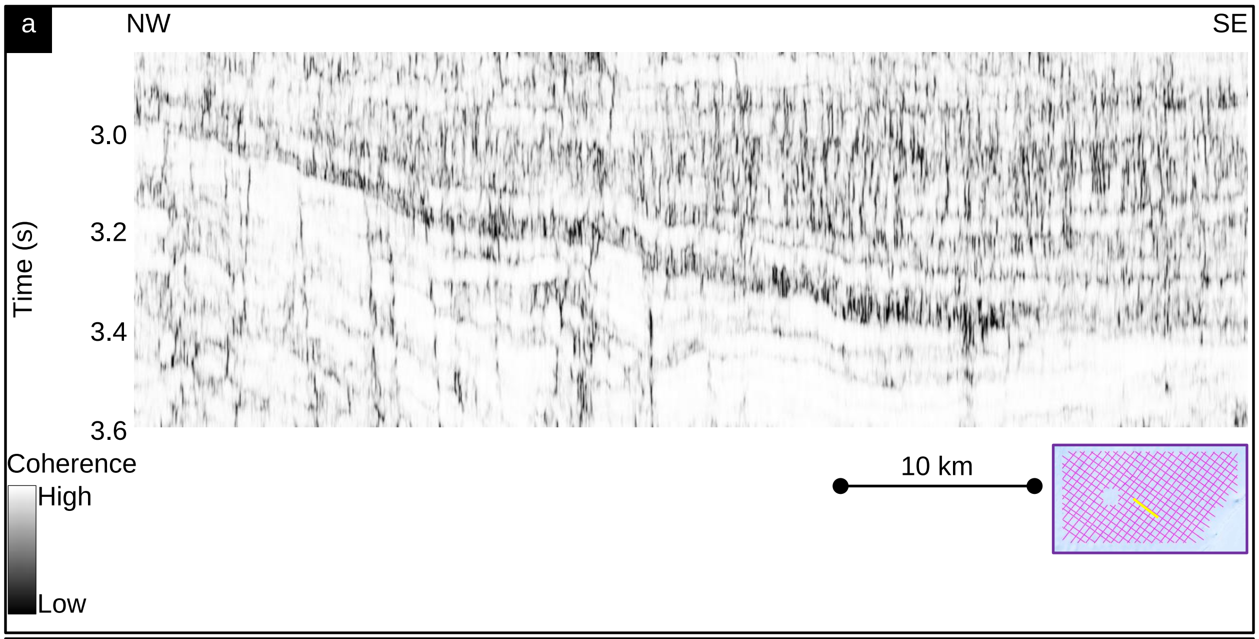


Figure 3.10 and Figure 3.11 show energy-ratio-similarity (a) before and (b) after preconditioning. Note the salt-and-pepper internal structure of MTCs are lost, but the boundary between chaotic and continuous facies are much better defined. Figure 3.12 and Figure 3.13 show the SOM result (a) with out and (b) with attribute preconditioning, in the same portions of strike and dip lines as

those shown in



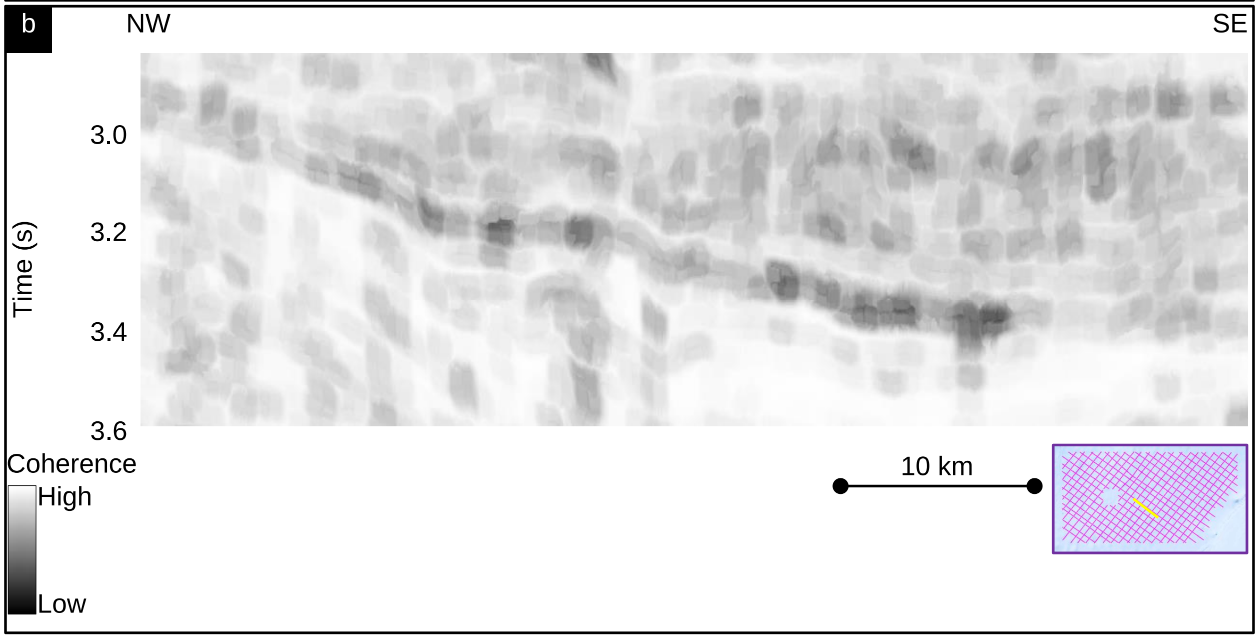


Figure 3.10 and Figure 3.11. The same salt-and-pepper effect is observed without attribute preconditioning, and thus the quality of SOM classification is significantly improved after attribute preconditioning.

Interpretation

Figure 3.14 shows a 2D seismic line with interpreted faults, key formation tops, MTCs, and channels. Major faults terminate against the top of the Muderong Shale, consistent with post-rifting subsidence during the middle Cretaceous. Figure 3.15 of co-rendered seismic amplitude with energy-ratio similarity attribute shows the chaotic, low-coherence nature of MTCs within the Muderong Shale. These MTCs can be tens-of-kilometers wide and are present in both the Muderong Shale and the shallower Tertiary carbonate sequences. For channels, I follow the display scheme described by Wallet (2016) to co-render seismic amplitude with structural curvatures to highlight channels' axes and levees (Figure 3.16). These channels have thicknesses ranging from 20 to 100 ms (equivalent to ~ 25-125 m for a velocity of 2500 m/s) and half-widths ranging from 0.5 to 1.5 km.

I find some local anomalies in the SW and NE corner of the seismic survey. Figure 3.17 is a 3D perspective view showing two perpendicular seismic lines near the SW corner of the study area, with two amplitude anomalies marked by yellow arrows. These anomalies are about 2 km long, exhibiting strong negative amplitude, anticlinal shape, and a lower frequency spectrum than the surrounding reflectors. Based on such characteristics, I interpret the two anomalies as bright spots that could be potential gas-charged reservoirs. Figure 3.18 is another 3D perspective view showing two perpendicular seismic amplitude profiles near the NE corner of the study area, with several anomalies marked by pink arrows. These anomalies are dome-shaped features, exhibiting velocity "pull-up" effect from their tops to the deeper section. Their characteristics are consistent with carbonate mounds around the world. Since these anomalies are within the thick Muderong Shale (which is a good seal), they could be potential reservoirs as well, if they are filled with hydrocarbons.

One way to obtain information regarding the depositional environment at a specific geological time is to generate a map of reflector convergence about a horizon of interest. Reflector convergence is an attribute that shows where reflectors are converging (pinching out) rather than parallel (conformal), which can aid in seismic stratigraphic analysis. Figure 3.19 shows a map of co-rendered reflector convergence magnitude and azimuth extracted around the top of the Muderong Shale formation. Generally, the layers are thinning toward the NW (green and cyan color), with some exception in the middle and eastern part of the study area where they are thinning toward the NE (purple color). This geometry indicates the presence of a major landmass in the NW of the North Carnarvon Basin, as well as some local structural highs in the center and NE part of the basin during the late Cretaceous.

After combining seismic attributes using the SOM algorithm, I co-render the result with seismic amplitude as shown in Figure 3.20 and Figure 3.21. I observe that red, light cyan, light purple, and blue colors correspond to chaotic portion of MTCs. Yellow and orange colors correspond to lower amplitude, conformal reflectors. Bright green color corresponds to moderate amplitude, continuous reflectors. Bright purple and dark violet correspond to high amplitude, continuous reflectors. To illustrate the chaotic portion of MTCs exclusively, I keep only red, light cyan, light purple, and blue colors, while setting all other colors transparent. The result is a co-rendered image of seismic amplitude and MTC "clusters" (Figure 3.22 and Figure 3.23), with occasional imperfections in which some bow-tie artifacts associated with faults are marked (yellow ellipses). Finally, I extract these MTC clusters from a set of 2D lines in my data. The size, shape, and internal structure of a gigantic MTC sequence can be observed via 3D visualization of the extracted geobodies (Figure 3.24). This MTC sequence is located at the SE part of the survey, in the bottom of the Tertiary carbonate sequences, extending more than 50 km in the dip direction and 80 km in the strike direction.

Pitfalls and Workarounds

Applying 3D interpretation workflow to 2D seismic lines is not a straightforward process. In this section, I identify several pitfalls I encountered in my study, as well as my workarounds to circumvent the incompatibility between 2D data and modern interpretation software.

The order of filters in data conditioning

There are two main steps in my data conditioning workflow: spectral balancing and structure-oriented filtering. Because they are both nonlinear filters, the order of application makes a difference. To evaluate the differences, I apply two data conditioning workflows (Figure 3.25) to a 2D seismic line. I find artifacts around faults when I apply structure-oriented filtering first, followed by spectral balancing (Figure 3.26). The other workflow does not generate such artifacts. This is because structure-oriented filter is an edge-preserving filter. For vertical faults, there is no change in the spectrum of the vertical trace. For dipping faults, however, the edge preservation introduces both high and low frequencies into the spectra of the vertical traces. These components lead to artifacts in spectral balancing. Therefore, the correct order of filters in data conditioning workflow is spectral balancing first, followed by structure-oriented filtering.

Reflector misties at line intersections

Figure 3.27 shows a 3D perspective view toward the intersection of two perpendicular seismic lines near the center of the study area. Migrated reflectors often do not tie at line intersections, making it difficult to pick a horizon. In the shallow part (marked by a yellow ellipse), reflectors on the right appear to arrive later than those on the left. However, in the deeper part (marked by an orange ellipse), the exact opposite phenomenon is observed: reflectors on the right appear to arrive sooner than those on the left. Those misties cannot be fixed by a simple time shift. Sattlegger and Egbers (1987) developed a workflow to properly map horizons picked on time-migrated 2D lines. They first picked horizons on the time-migrated 2D lines as good as possible, depth-converted the time-migrated 2D lines, performed modeling, and repeated the process until the horizons tie at line intersections. Next, they interpolated the horizons to form maps, and then migrate the maps using a process called "3D map migration." Unfortunately, their workflow requires a velocity model for depth-conversion, which is not available to me. My workaround is to construct two separate horizon picks for each physical surface. First, I perform horizon picking where the lines tie. Then, I make two copies of the common picks and continue to define two separate sets of horizon picks: one parallel to the lines trending NW-SE, and one parallel to the lines trending NE-SW (Figure 3.28). Attribute extracted along both sets of horizons can then be interpolated to form attribute maps.

Vector attribute extraction along a surface

Among the modern interpretation software packages I tested, none supports attribute extraction from 2D data along a surface. Thus, I need to develop my own tool to extract seismic attributes at the intersections between an interpolated surface and the 2D lines. For vector attributes, such as reflector convergence, the seismic line's direction of increasing CDP number is important. I need to make sure all lines parallel to a trend have the same direction of increasing CDP number. Each line trend should then have a separate set of extracted data points (Figure 3.29). Next, I generate surfaces from those sets of points (Figure 3.30) and combine them into a co-rendered magnitude and azimuth map (Figure 3.19) using the following equations:

$$convergence\ magnitude = \sqrt{convergence_{NW}^2 + convergence_{NE}^2} \tag{3.1}$$

$$convergence\ azimuth = \arctan\left(\frac{convergence_{NE}}{convergence_{NW}}\right) + NW\ azimuth \tag{3.2}$$

Where

$convergence_{NW}$ is the reflector convergence component along NW-trending lines,

$convergence_{NE}$ is the reflector convergence component along NE-trending lines, and

$NW\ azimuth$ is the azimuth of NW-trending lines.

Geobody extraction from 2D data

To visualize SOM clusters in 3D, I first attempt to use a geobody extraction tool that works well on 3D data. However, almost all geobody extraction engines assume 3D data inputs, thus making it impossible to extract or display geobodies from 2D data using modern interpretation software. My next option is to use color transparency to highlight only those SOM clusters of interest on several 2D lines and then display those lines in 3D. However, I encountered a graphical shortcoming in which the line farthest from the viewpoint is sometimes not rendered correctly at some angle of view (Figure 3.31). This might be due to the interpretation software's internal graphical engine: some pixels are assumed to be unnecessary and therefore not rendered on the computer screen in order to reduce memory usage. Therefore, I decided to program my own tool to extract data points of specific values from multiple 2D lines. With this tool, I am able to extract SOM clusters corresponding to MTCs and display them in a 3D viewport (Figure 3.24).

Conclusions

By applying data conditioning to the 2D data, I am able to improve seismic image quality. However, I need to consider the order of steps in data conditioning workflow and apply spectral balancing first, followed by structure-oriented filtering, in order to avoid unwanted artifacts. Seismic attributes, including coherence, envelope, and structural curvature, help me interpret regional geological features (such as the wide-spread Muderong Shale and the channels within the Tertiary carbonate sequences), as well as local anomalies (such as bright spots and carbonate mounds). Analysis of seismic facies, including MTCs and channels, can be accelerated using Self-Organizing-Map classification. Nevertheless, using modern interpretation software, I can neither extract attributes along a surface nor display geobodies from 2D data. Therefore, I need to develop my own point-by-point attribute extraction tools in order to correctly extract and display attribute data points from 2D lines.

Acknowledgments

I thank Schlumberger for a license to the Petrel interpretation software for use in research and education. Special thanks to Geoscience Australia for providing the data used in this project. Financial support was through the industry sponsors of the OU Attribute-Assisted Seismic Processing and Interpretation (AASPI) consortium.

Appendix A: Data Conditioning Workflow Parameters

This appendix lists parameters of spectral decomposition (Table 3.A-1), structural dip computation (Table 3.A-2), dip filtering (Table 3.A-3), coherence computation (Table 3.A-4), and structure-oriented filtering (Table 3.A-5).

Appendix B: Attribute Computation Parameters

This appendix lists parameters of curvature and reflector convergence computation (Table 3.B-1) and GLCM attribute computation (Table 3.B-2). Parameters for spectral attribute (such as peak frequency) are the same as in Table 3.A-1. Parameters for coherence attributes (such as energy-ratio similarity and coherent energy) are the same as in Table 3.A-4.

Appendix C: SOM Workflow Parameters

This appendix lists parameters of smoothing operation (Table 3.C-1), Kuwahara filtering (Table 3.C-2), and SOM classification (Table 3.C-3). Attribute preconditioning flowchart is shown in Figure 3.C-1.

References

Bahorich, M. S. and S. R. Bridges, 1992, Seismic sequence attribute map (SSAM): 62nd Annual International Meeting of the SEG, Expanded Abstracts: 227-230. doi: 10.1190/1.1822047

Chongzhi, T., B. Guoping, L. Junlan, D. Chao, L. Xiaoxin, L. Houwu, and W. Dapeng, 2013, Mesozoic lithofacies palaeogeography and petroleum prospectivity in North Carnarvon basin, Australia: Journal of Palaeogeography, **2**, 81-92. doi: 10.3724/SP.J.1261.2013.00019

Chopra, S., and K. J. Marfurt, 2007, Seismic attributes for prospect identification and reservoir characterization: SEG Geophysical Development Series, **11**, 45-72. doi: 10.1190/1.9781560801900.ch3

Hutchinson, B., 2016, Application and Limitations of Seismic Attributes on 2D Reconnaissance Surveys: Master's thesis, University of Oklahoma.

Hutchinson, B., J. Qi, F. Li, O. Olorunsola, and K. J. Marfurt, 2016, Pitfalls in poststack data conditioning: Distinguishing enhanced geology from enhanced artifacts. 86th Annual International Meeting of the SEG, Expanded Abstracts, 1783-1787. doi: 10.1190/segam2016-13970185.1

Lin, T., T. Ha, K. J. Marfurt, and K. L. Deal, 2016, Quantifying the significance of coherence anomalies: Interpretation, **4**, T205-T213. doi: 10.1190/INT-2015-0102.1

Magee, C., C. A. L. Jackson, and N. Schofield, 2013, The influence of normal fault geometry on igneous sill emplacement and morphology: Geology, **41**, 407–410. doi: 10.1130/G33824.1

Mai, H. T., K. J. Marfurt, and A. S. Chavez-Perez, 2009, Coherence and volumetric curvatures and their spatial relationship to faults and folds, an example from Chicontepec Basin, Mexico: 79th Annual International Meeting of the SEG, Expanded Abstracts, 1063-1067. doi: 10.1190/1.3255033

Marfurt, K. J., 2006, Robust estimates of reflector dip and azimuth: Geophysics, **71**, 29–40. doi: 10.1190/1.2213049

Marfurt, K. J., and J. Rich, 2010, Beyond curvature – Volumetric estimation of reflector rotation and convergence: 80th Annual International Meeting of the SEG, Expanded Abstracts, 1467–1472. doi: 10.1190/1.3513118

Marfurt, K. J., 2018, Seismic Attributes as the Framework for Data Integration throughout the Life of the Oilfield: SEG Distinguished Instructor Series, 25-149. doi: 10.1190/1.9781560803522.ch2

Matos, M., M. Yenugu, S. M. Angelo, and K. J. Marfurt, 2011, Integrated seismic texture segmentation and cluster analysis applied to channel delineation and chert reservoir characterization: Geophysics, **76**, 11-21. doi: 10.1190/geo2010-0150.1

McArdle, NJ., D. Iacopini, M. A. KunleDare, and G. S. Paton, 2014, The use of geologic expression workflows for basin scale reconnaissance: A case study from the Exmouth Subbasin, North Carnarvon Basin, northwestern Australia: Interpretation, **2**, 163-177. doi: 10.1190/INT-2013-0112.1

Mihut, D., and R. D. Muller, 1998, Volcanic margin formation and Mesozoic rift propagators in the Cuvier Abyssal Plain off Western Australia: Journal of Geophysical Research, **103**, 27135–27149. doi: 10.1029/97JB02672

Qi, J., T. Lin, T. Zhao, F. Li, and K. J. Marfurt, 2016, Semisupervised multiattribute seismic facies analysis: Interpretation, **4**, SB91-SB106. doi: 10.1190/INT-2015-0098.1

Sattlegger, J. W. and H. Egbers, 1987, Three-dimensional mapping of horizons picked on two-dimensionally migrated seismic sections: Exploration Geophysics, **18**, 189-192. doi: 10.1071/EG987189

Smith, M., 2014, Cenozoic stratigraphy of the North Carnarvon Basin: Insights for the growth history of carbonate margins of the North-West Shelf: Master’s thesis, University of Western Australia.

Tellez Rodriguez, J. J. J., 2015, Seismic-sequence stratigraphy framework and architectural elements for Cenozoic strata at the Rankin Platform sub basin, North Carnarvon Basin, Australia: Master’s thesis, University of Oklahoma.

Wallet, B. C., 2016, Attribute expression of channel forms in a hybrid carbonate turbidite formation: Interpretation, **4**, SE75-SE86. doi: 10.1190/INT-2015-0108.1

Zhao T., J. Zhang, F. Li, and K. J. Marfurt, 2016, Characterizing a turbidite system in Canterbury Basin, New Zealand, using seismic attributes and distance-preserving self-organizing maps: Interpretation, **4**, SB79-SB89. doi: 10.1190/INT-2015-0094.1

Figures

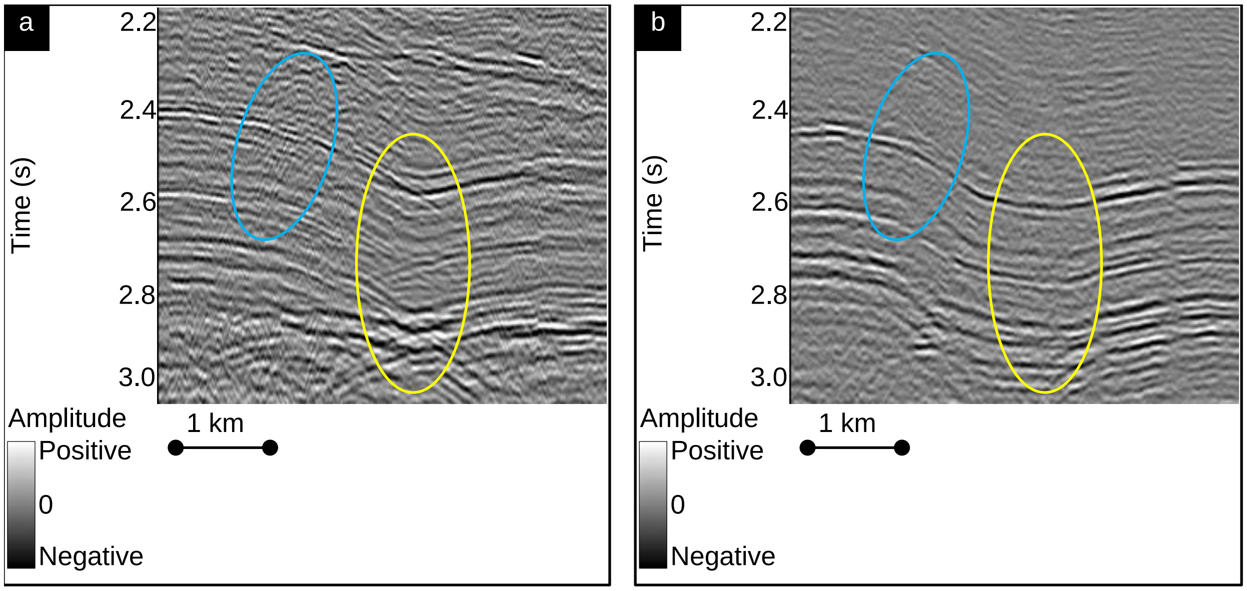


Figure 3.1. A portion of (a) a 2D line and (b) the equivalent vertical profile through a 3D data covering the same area, showing a syncline (Modified from Hutchinson, 2016). Note the cross-cutting artifacts (cyan ellipse) and bow-tie artifacts (yellow ellipse) present in (a) but absent in (b).

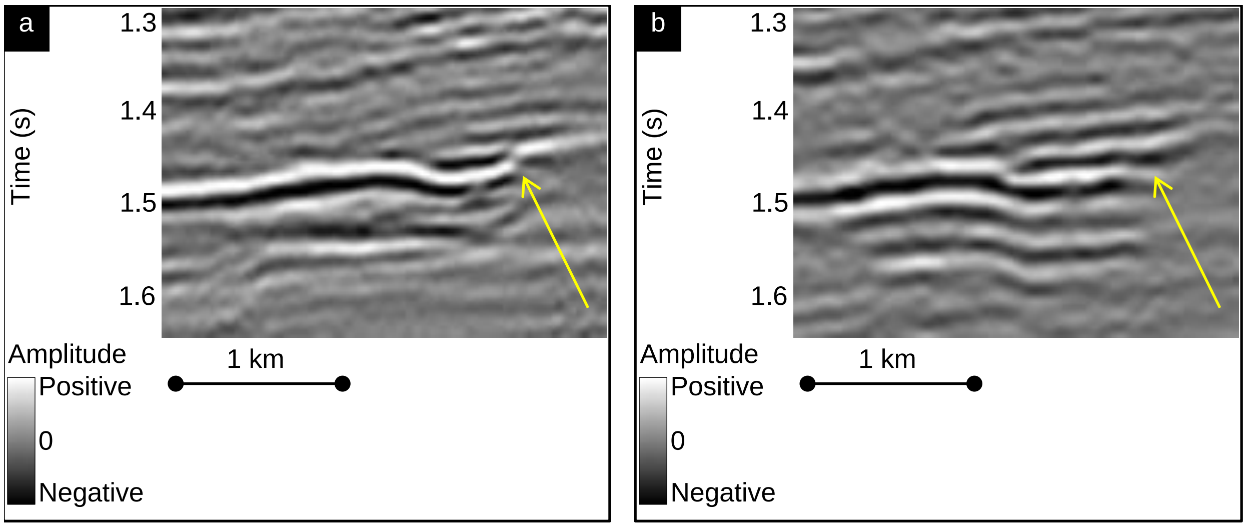


Figure 3.2. A portion of (a) a 2D line and (b) the equivalent vertical profile through a 3D data covering the same area, showing a channel perpendicular to the plane of view (Modified from Hutchinson, 2016). Note how the right edge of this channel (yellow arrow) appears continuous in (a) but discontinuous in (b).

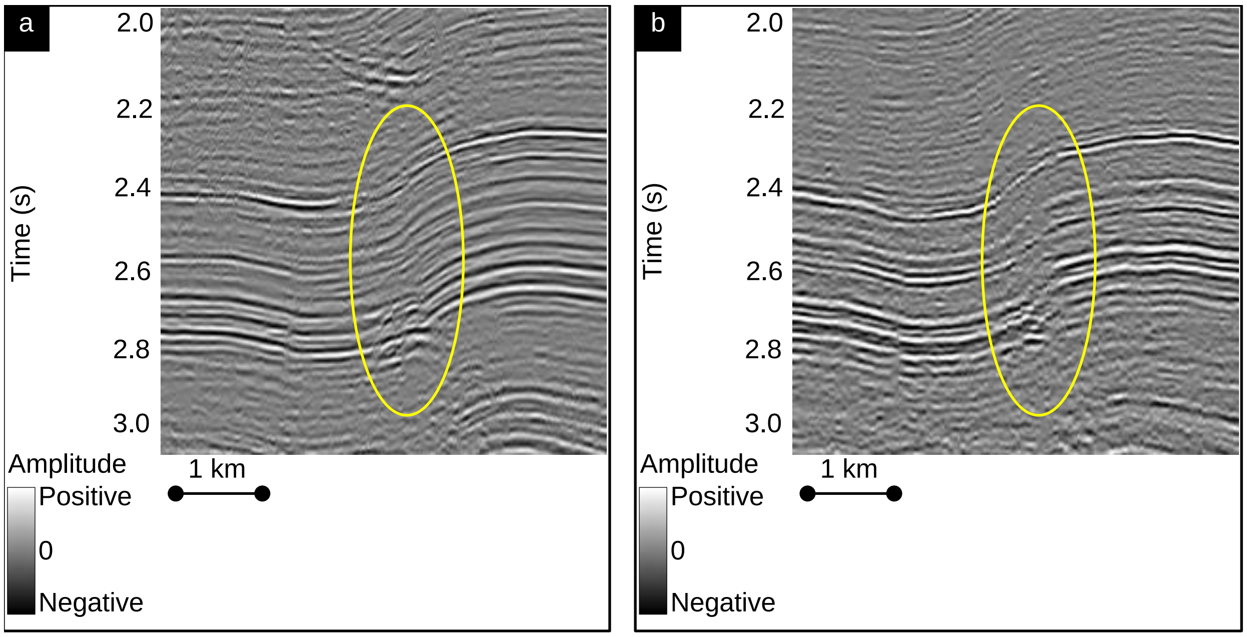


Figure 3.3. A portion of (a) a 2D line and (b) the equivalent vertical profile through a 3D data covering the same area, showing a fault-propagation fold (Modified from Hutchinson, 2016). Note the reflectors on the left flank of this fold (yellow ellipse) cross-cut each other in (a) but are correctly imaged in (b). The 3D data show that the 2D line is not perpendicular to the strike of the fault-propagation fold. Events measured by the 2D line should be imaged out of the plane while events need to image this feature on the 2D line are not measured at all.

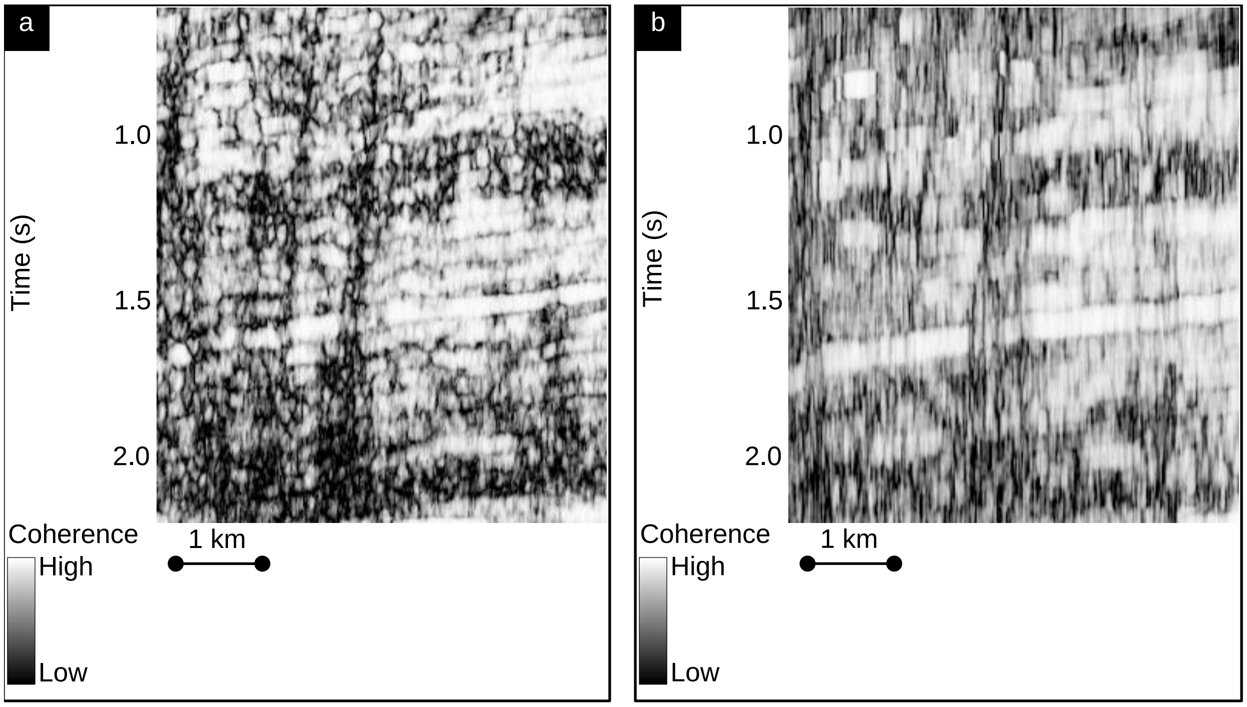


Figure 3.4. A portion of (a) a 2D coherence profile and (b) the equivalent vertical coherence profile through a 3D data covering the same area, showing faults (Modified from Hutchinson, 2016). The image in (b) appears smooth because of the large analysis window size used in coherence attribute calculation to overcome noise in 3D data. However, the same fault features can be observed in both (a) and (b).

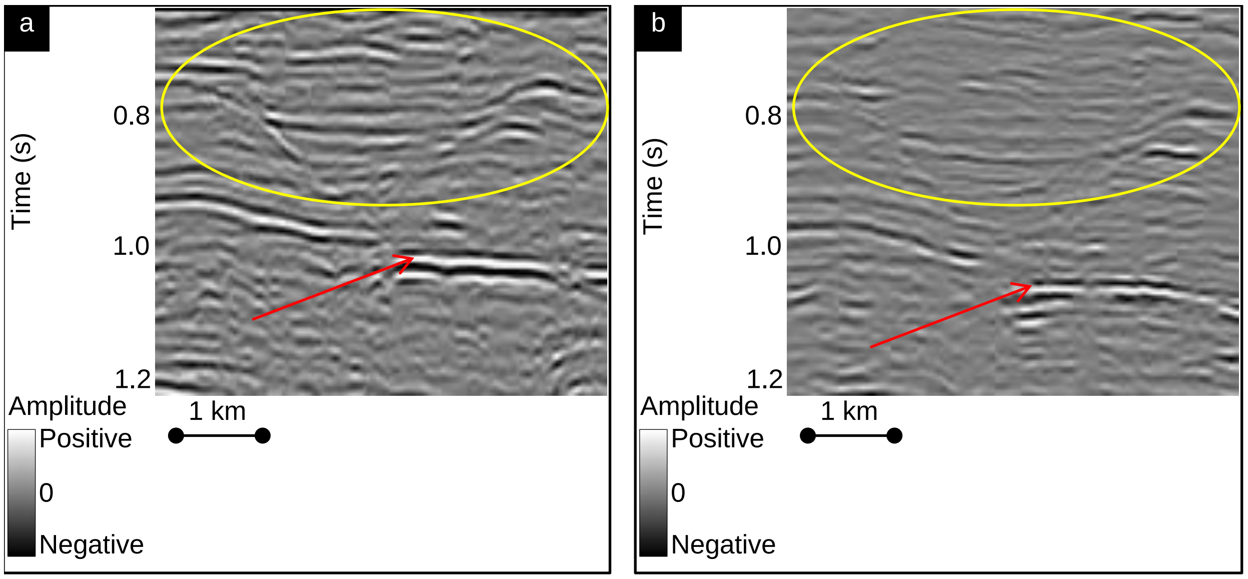


Figure 3.5. A portion of (a) a 2D line and (b) the equivalent vertical profile through a 3D data covering the same area, showing a shallow section around 1.0 s (Modified from Hutchinson, 2016). The color bars of (a) and (b) are scaled in such a way that the reflectors indicated by red arrow have the same brightness. In the yellow ellipse, the amplitude contrast between peaks and troughs in (a) are higher than (b). These differences are most likely due to higher fold at shallow depth in 2D seismic data.

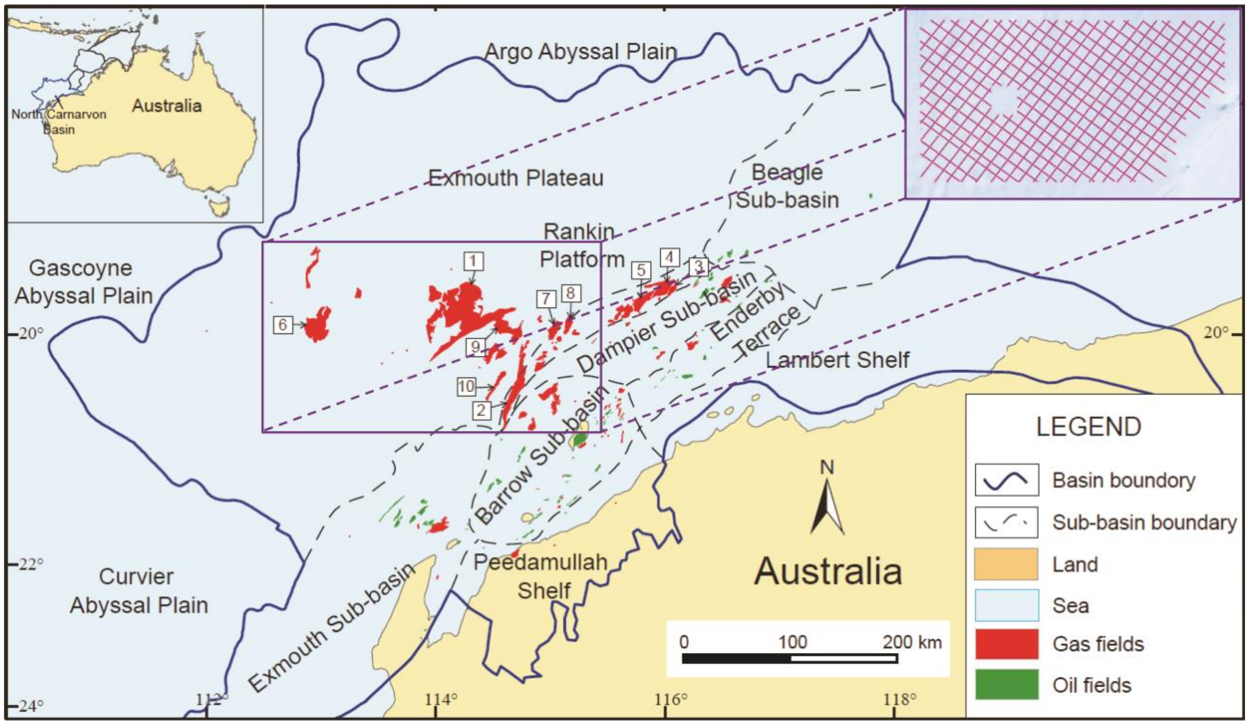


Figure 3.6. North Carnarvon Basin, Australia (Modified from Chongzhi et al., 2013). The study area (violet rectangle) consists of 55 2D seismic lines at the center of the Exmouth Plateau, where most of the major gas fields are located. Acquisition grid shown in upper-right corner of this figure.

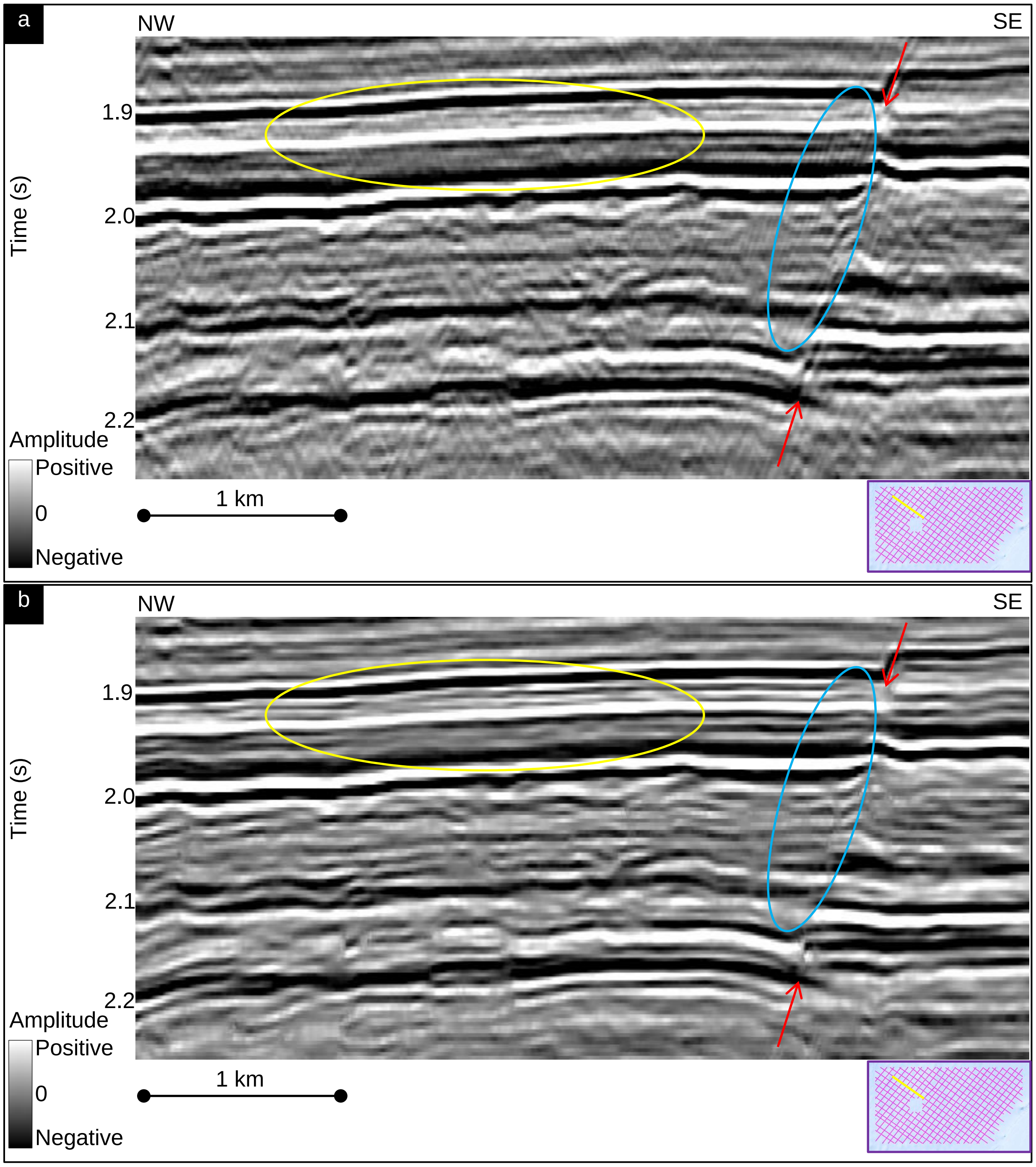


Figure 3.7. A shallow section of a 2D line (a) before and (b) after data conditioning (including spectral balancing and structure-oriented filtering). Reflectors exhibit broader bandwidth and are more continuous (yellow ellipse), faults are sharper (red arrows), and cross-cutting migration artifacts are suppressed (cyan ellipse).

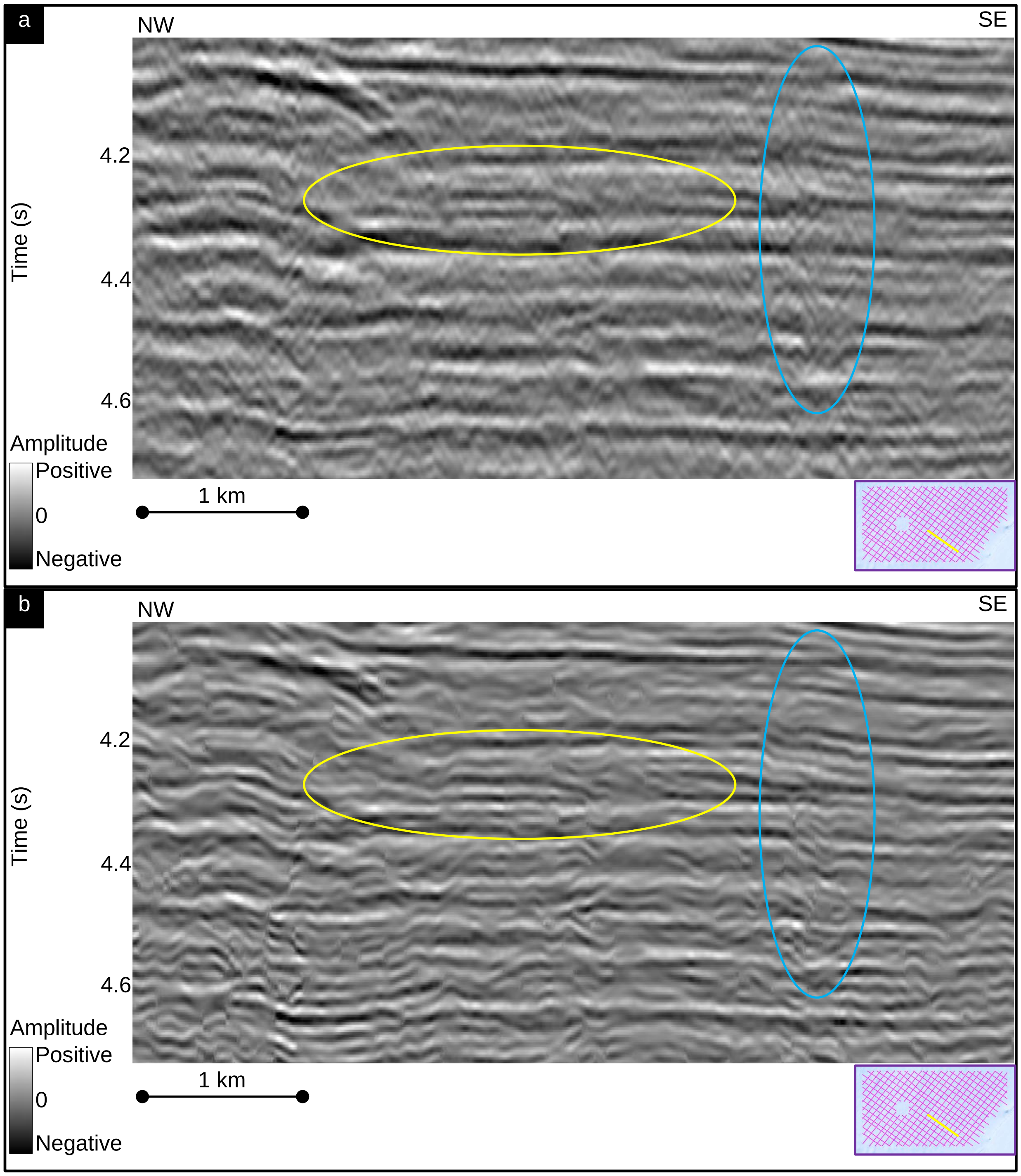


Figure 3.8. A deep section of a 2D line (a) before and (b) after data conditioning. A similar effect to Figure 3.7 is observed: reflectors are better resolved (yellow ellipses), and cross-cutting migration artifacts are suppressed (cyan ellipse).

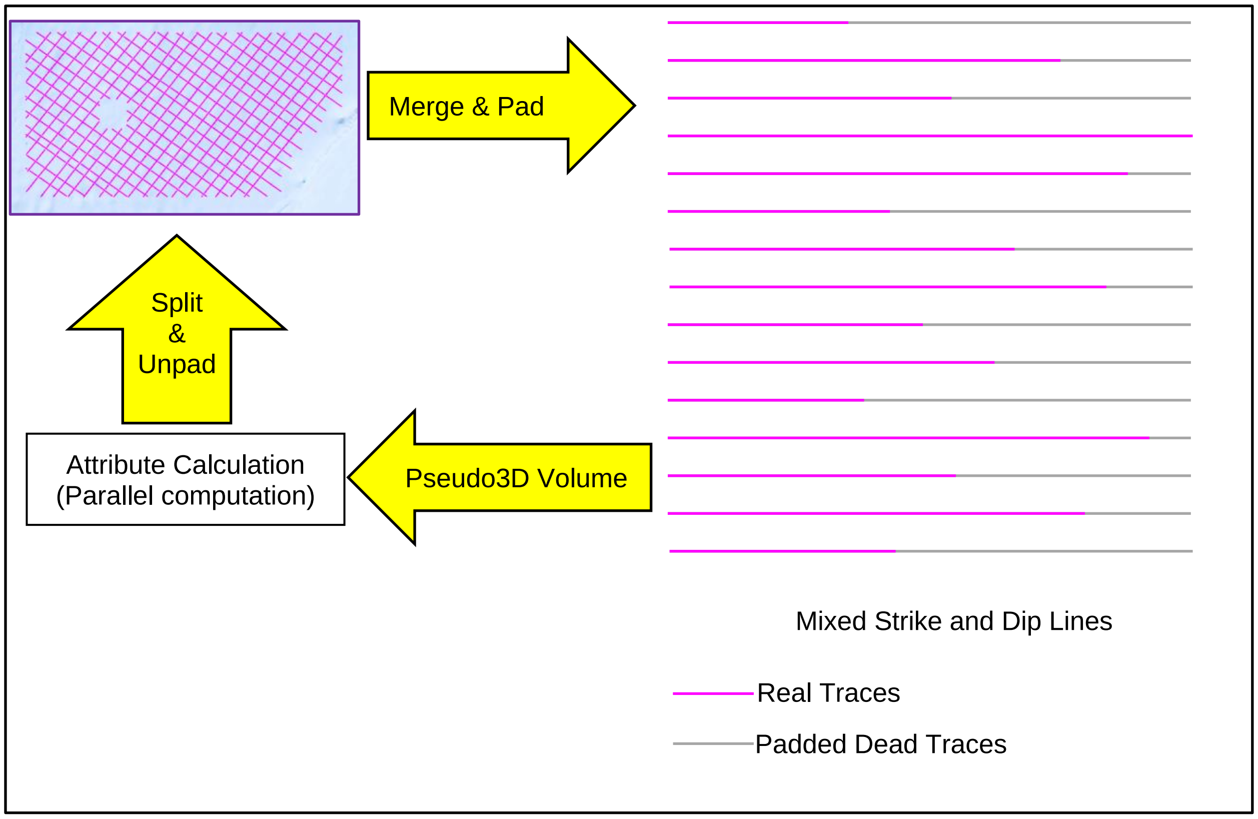

Merge & Pad
Split & Unpad
Attribute Calculation (Parallel computation)
Pseudo3D Volume
Mixed Strike and Dip Lines
Real Traces
Padded Dead Traces

Figure 3.9. Schematic diagram of my multi-line attribute calculation workflow. I pad multiple 2D seismic lines with dead traces to be of the same length as the longest line and then merge them into a pseudo3D volume. This pseudo3D volume is then used as the input to attribute calculators but with "crossline" analysis window size set to zero. After the computation, I delete the dead traces and reassign each "inline" back to the appropriate line in the 2D survey.

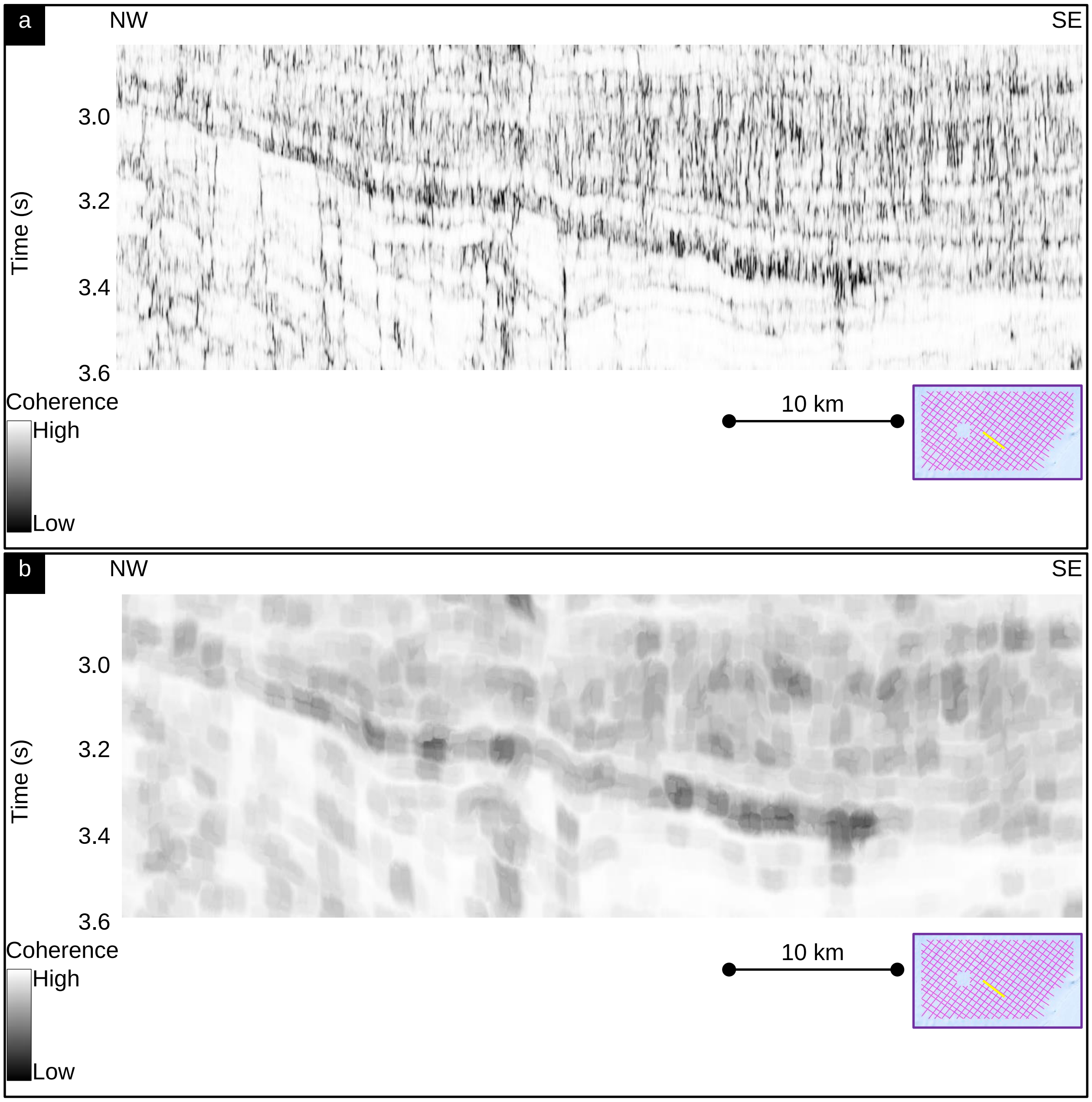
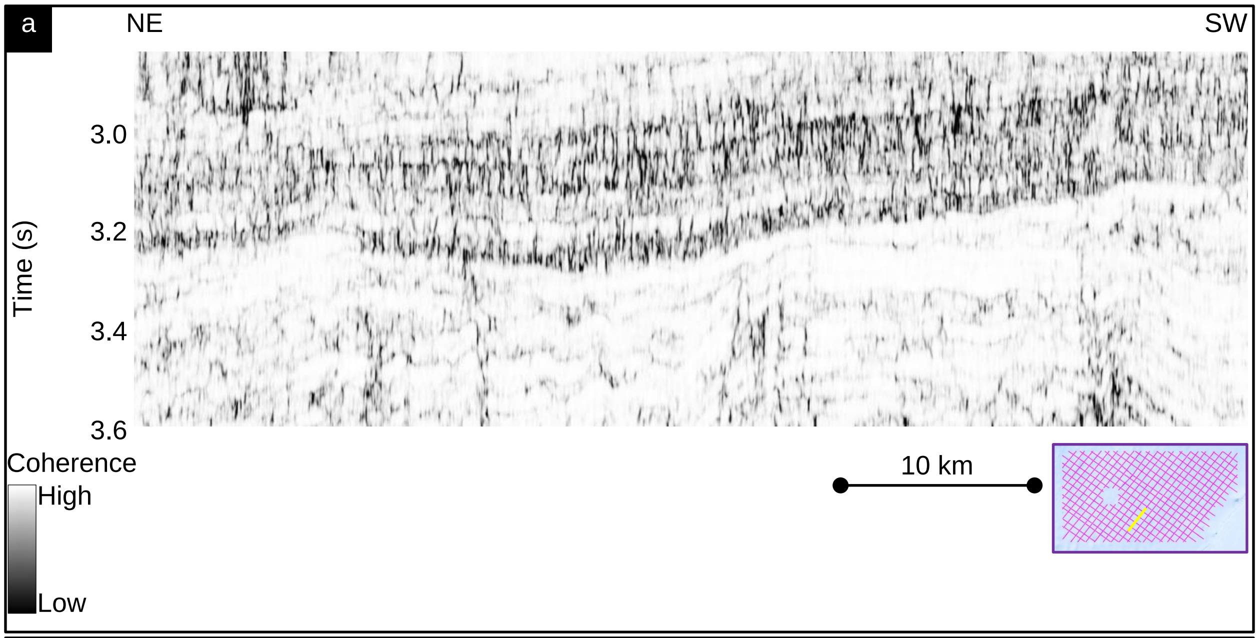


Figure 3.10. Energy-ratio-similarity attribute computed on a portion of a strike line (a) before and (b) after preconditioning. Note the salt-and-pepper internal structure of MTCs are lost, but the boundary between chaotic and continuous facies are much better defined.



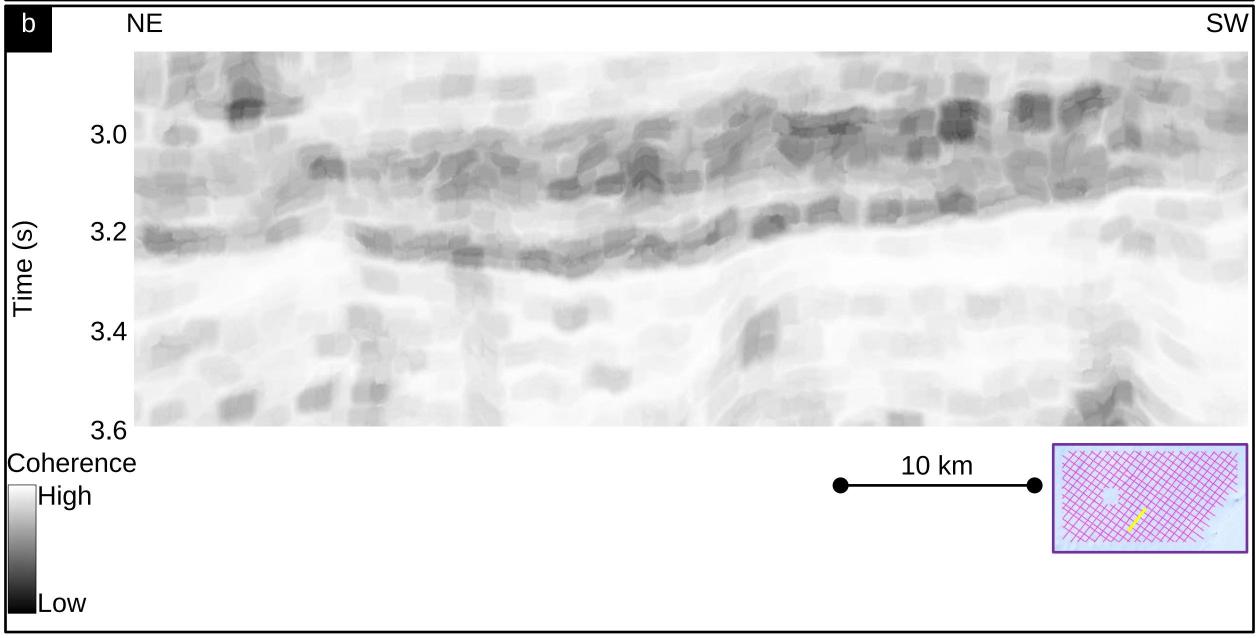
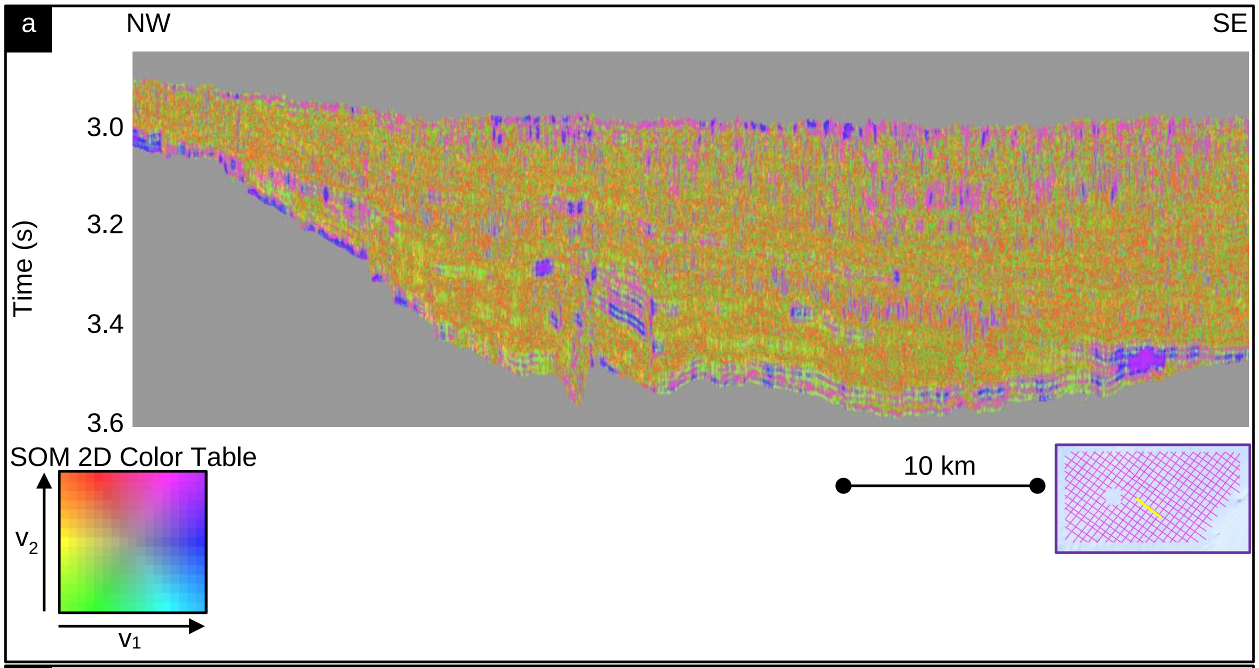


Figure 3.11. Energy-ratio-similarity attribute computed on a portion of a dip line (a) before and (b) after preconditioning. Note the salt-and-pepper internal structure of MTCs are lost, but the boundary between chaotic and continuous facies are much better defined.


a
NW
SE
3.0
3.2
3.4
3.6
Time (s)
SOM 2D Color Table
v_2
v_1
10 km

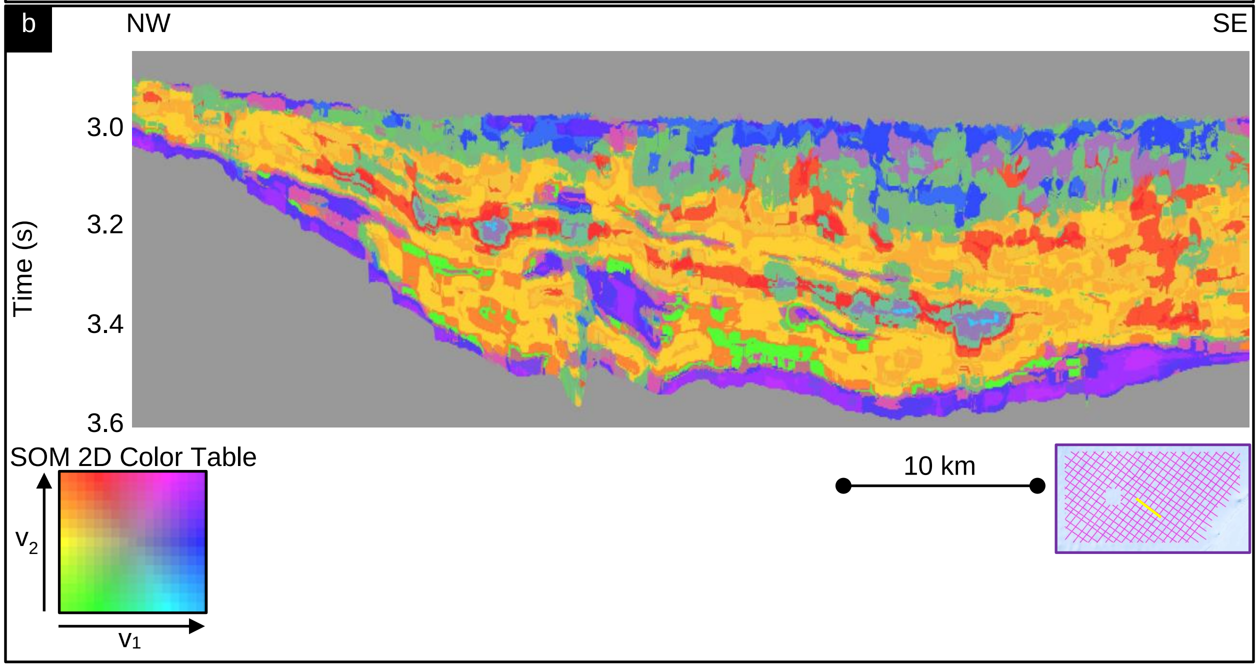
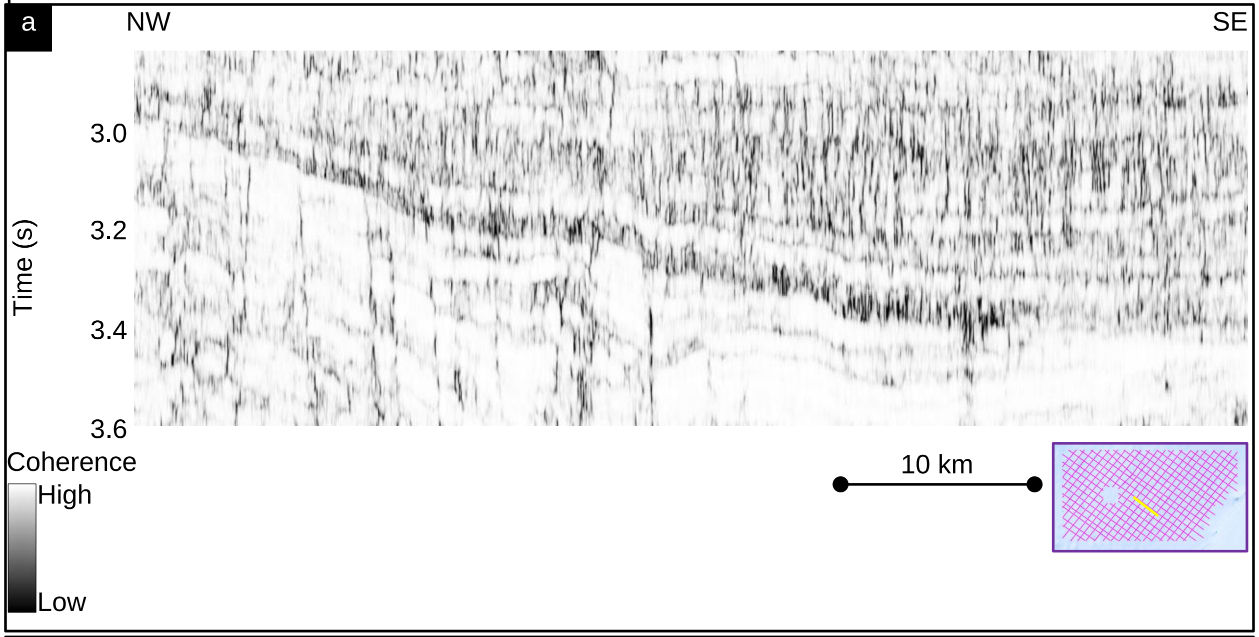

b
NW
SE
3.0
3.2
3.4
3.6
Time (s)
SOM 2D Color Table
v_2
v_1
10 km

Figure 3.12. SOM result (a) with out and (b) with attribute preconditioning, computed on the same portion of a strike line in



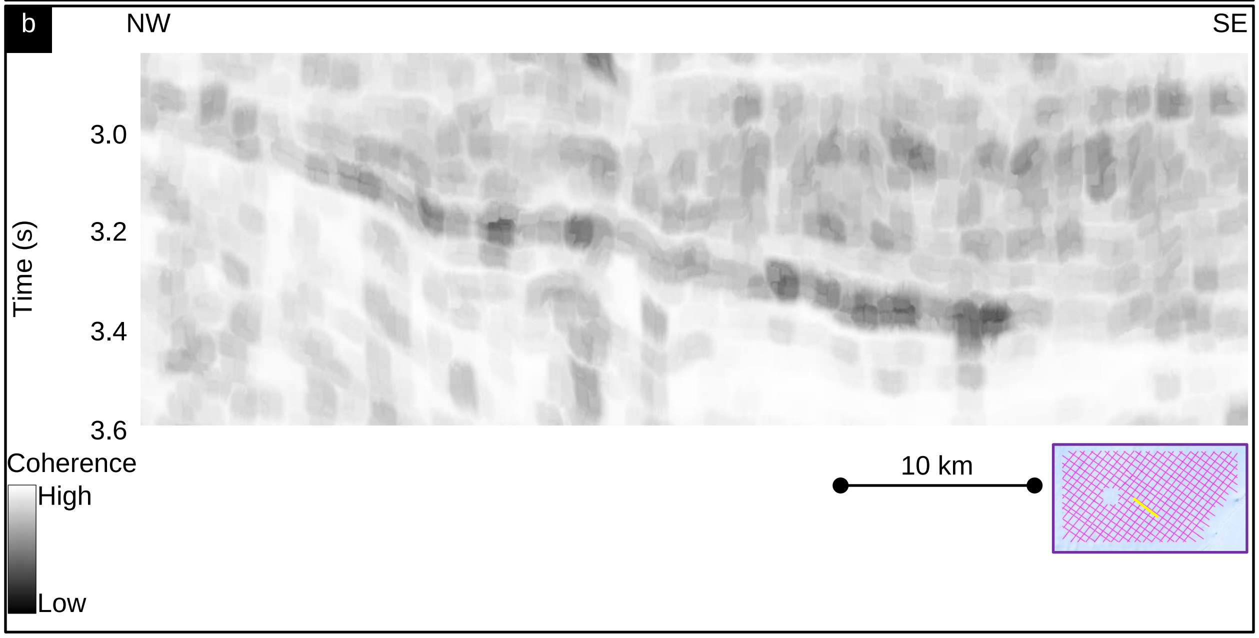


Figure 3.10. v_1 and v_2 are the first two eigenvectors of the four attributes used in SOM computation. The same salt-and-pepper effect is observed without attribute preconditioning. The quality of SOM classification in (b) is significantly higher than that in (a).

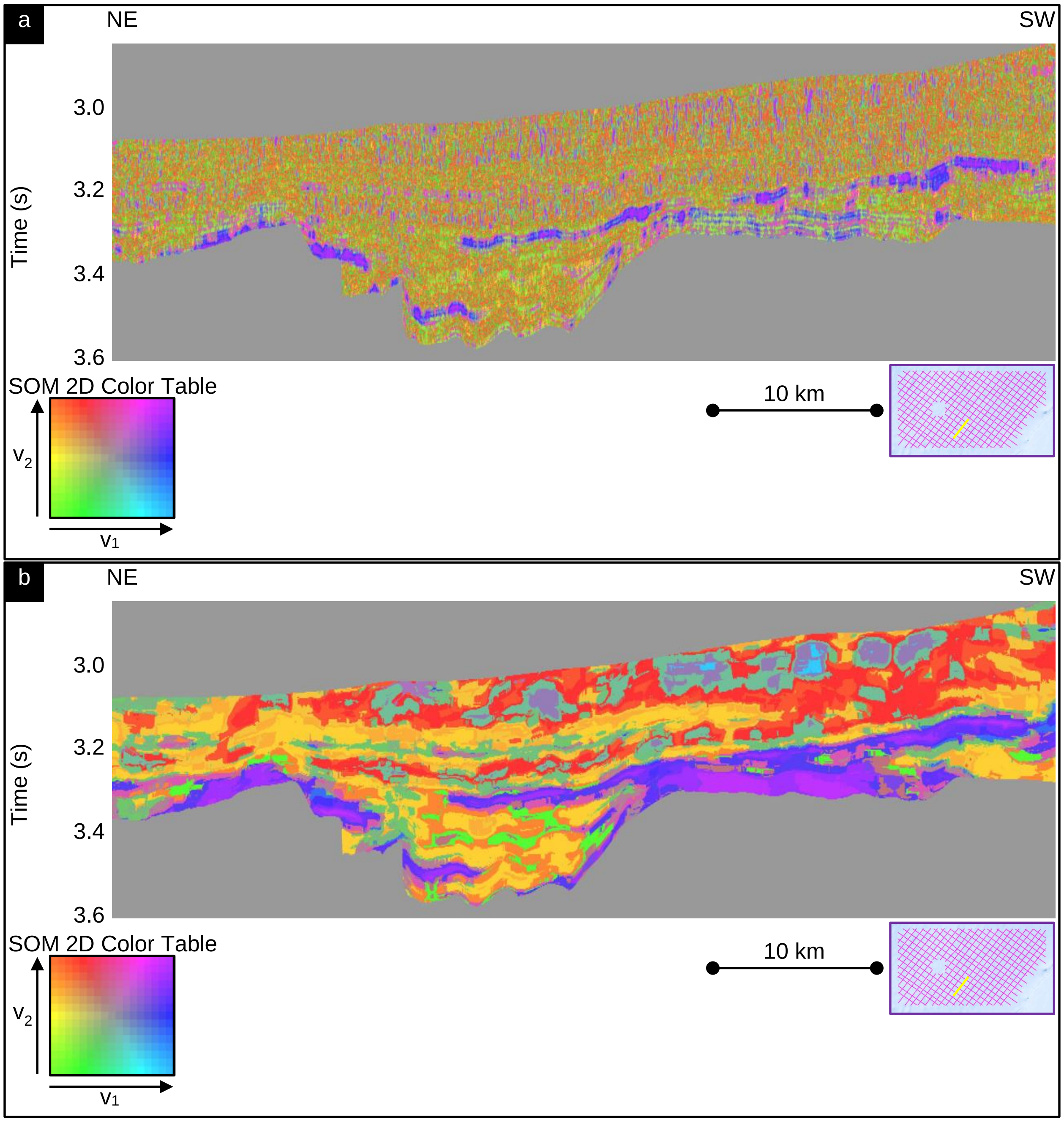


Figure 3.13. SOM result (a) with out and (b) with attribute preconditioning, computed on the same portion of a dip line in Figure 3.11. v_1 and v_2 are the first two eigenvectors of the four attributes used in SOM computation. The same salt-and-pepper effect is observed without attribute preconditioning. The quality of SOM classification in (b) is significantly higher than that in (a).

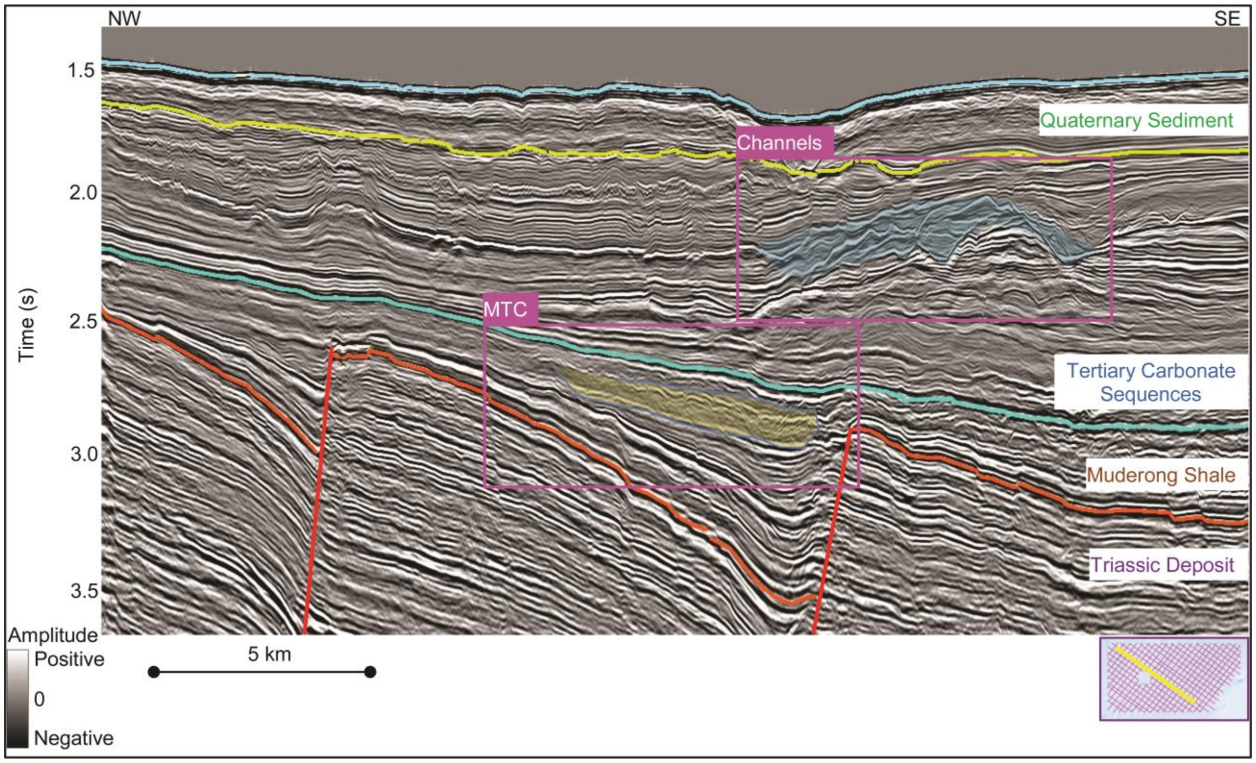


Figure 3.14. A representative portion of a 2D seismic line, with interpreted events. Major faults (red lines) terminate against the top of the Muderong Shale, consistent with post-rifting subsidence of the area during the middle Cretaceous. Mass Transport Complexes (MTCs) within the Muderong Shale and Tertiary carbonate are chaotic (Figure 3.15). Within the carbonate sequences, a wavy pattern of channels can be observed (Figure 3.16).

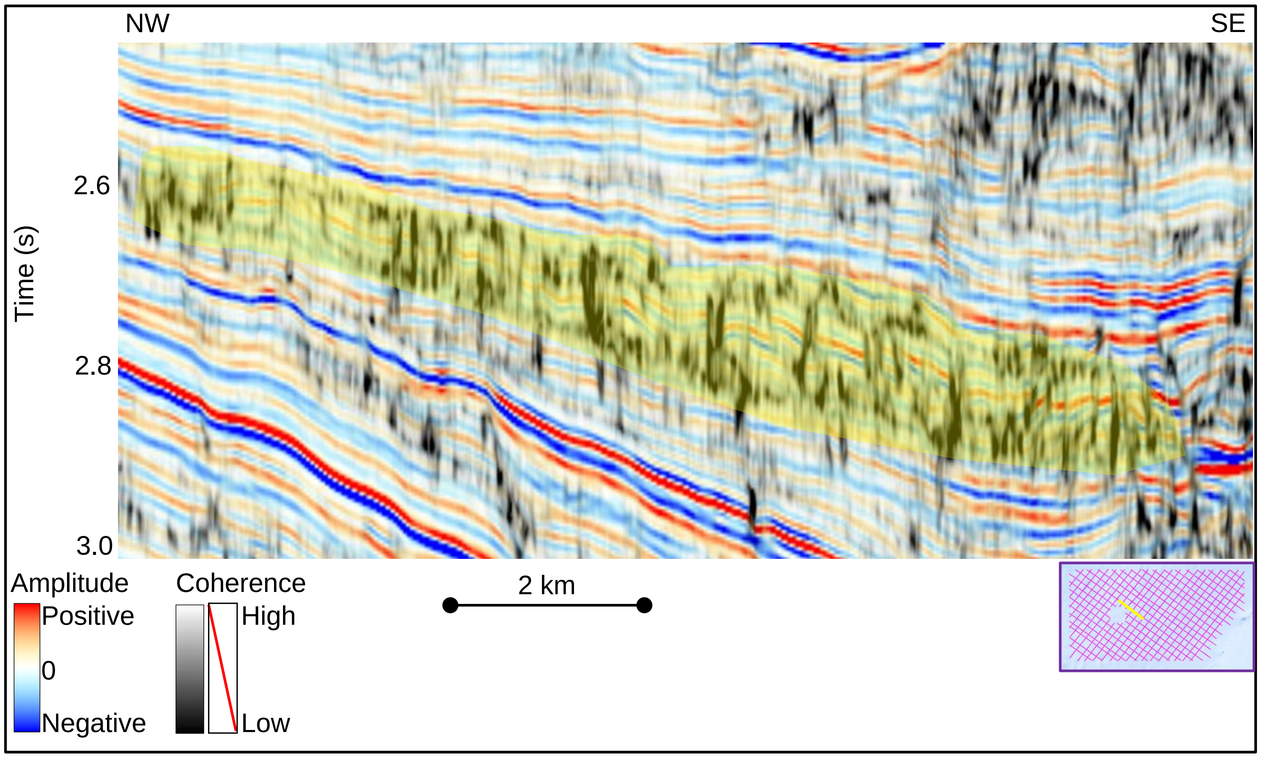


Figure 3.15. Co-rendered coherence and seismic amplitude profile of the region containing MTCs. Seismic amplitude is plotted using a red-white-blue color bar. These MTCs are chaotic, exhibits low coherence, and are tens-of-kilometers wide (yellow marker). We found MTCs both within the Muderong Shale and the Tertiary carbonate sequences.

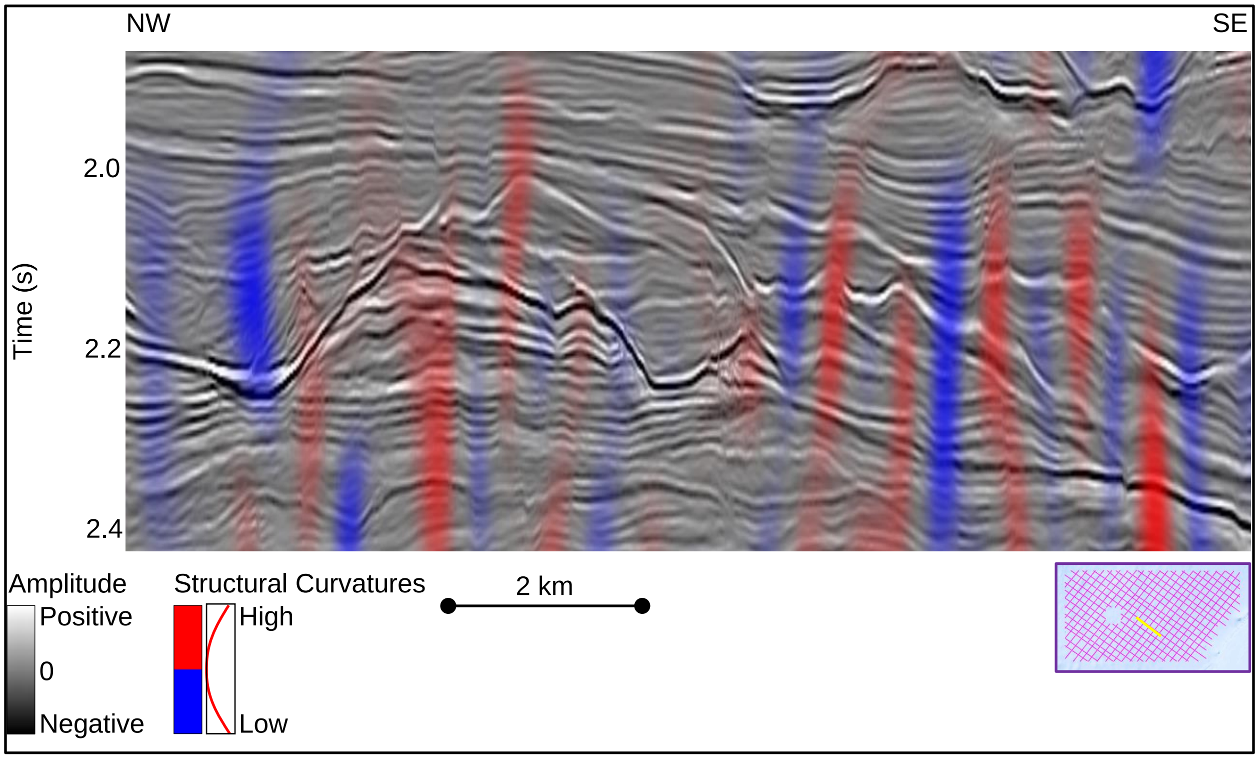


Figure 3.16. Co-rendered curvature and seismic amplitude profile of the region containing channels. The opacity curve is set so that only high absolute curvature values are highlighted. High negative curvature values correspond to channel axes, similar to Wallet's observations (2016). High positive curvature values correspond to channel levees. These channels have thicknesses ranging from 20 to 100 ms (equivalent to ~ 25-125 m for v=2500 m/s) and half-widths ranging from 0.5 to 1.5 km.

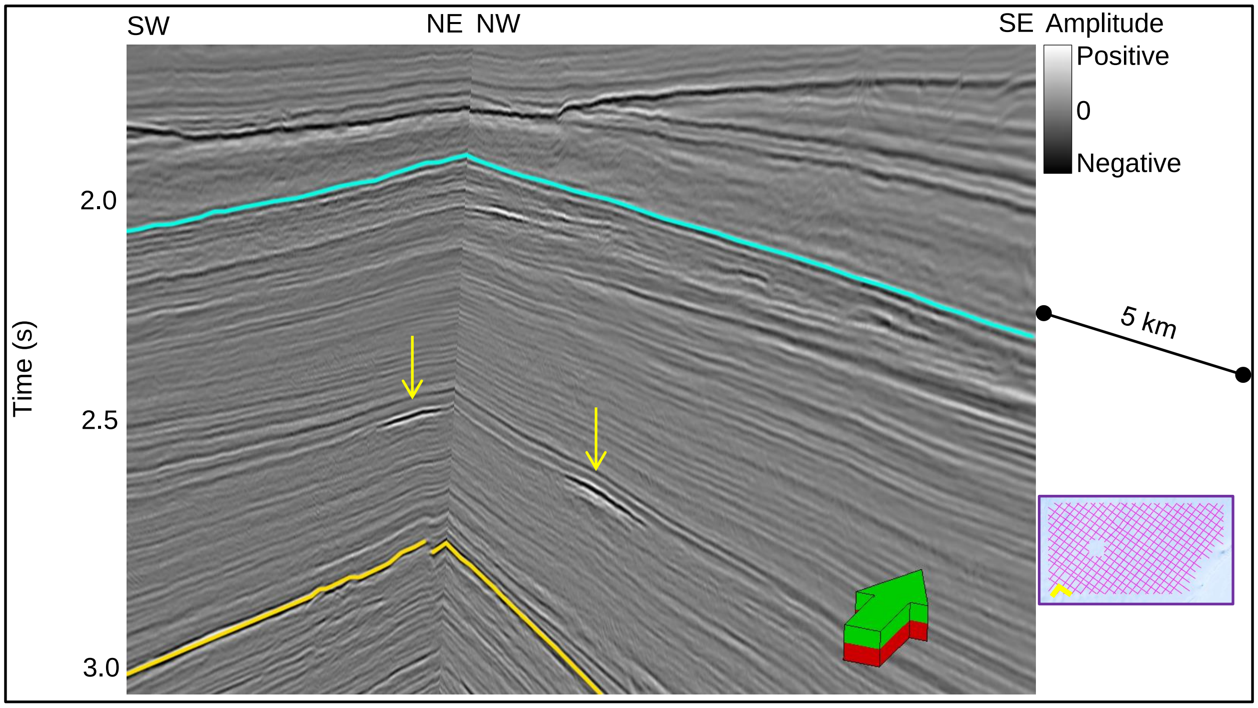


Figure 3.17. 3D perspective view showing two perpendicular 2D lines at the SW corner of the study area. Yellow arrows indicate two amplitude anomalies within the Muderong Shale. These anomalies are approximately 2-km long, exhibiting a stronger negative amplitude, more anticlinal shape, and lower frequency spectrum than the surrounding reflectors. I interpret these anomalies as bright spots that could be potential gas-charged reservoirs.

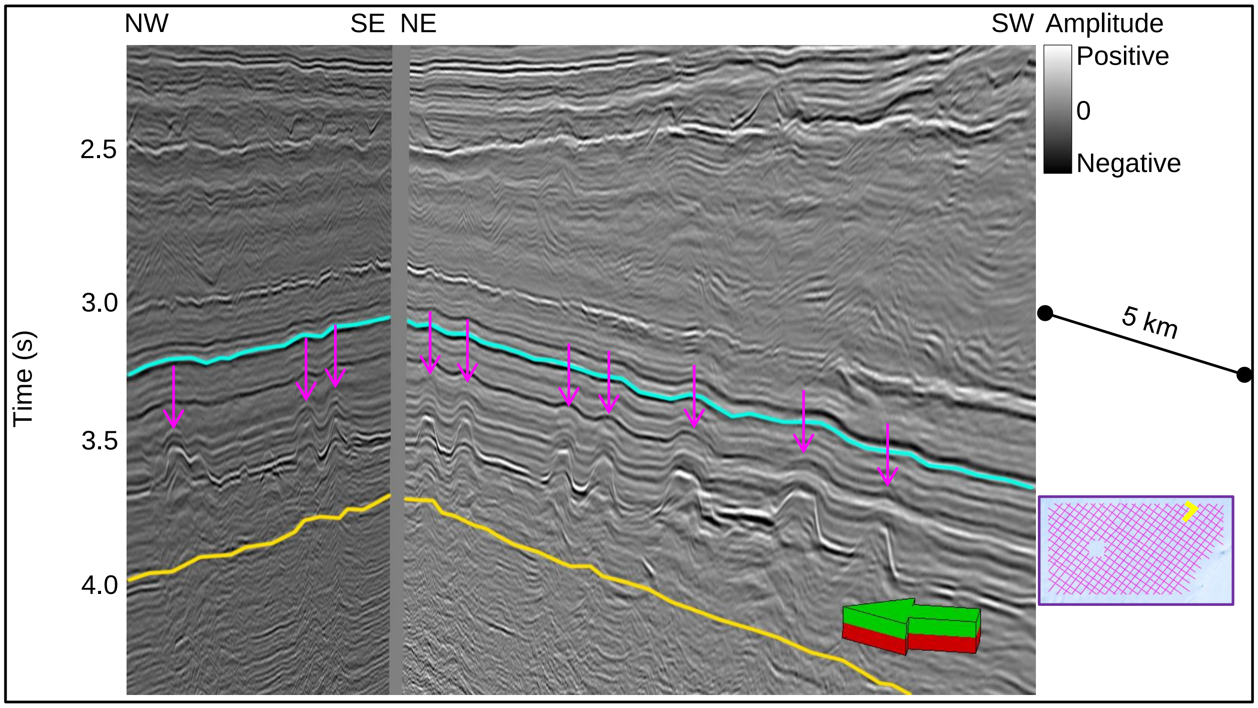


Figure 3.18. 3D perspective view showing two perpendicular 2D lines near the NE corner of the study area. Pink arrows indicate multiple dome-shaped features (~1 km wide) within the Muderong Shale. These domes are possibly carbonate mounds developed in the middle Cretaceous that could be potential reservoirs if filled with hydrocarbons.

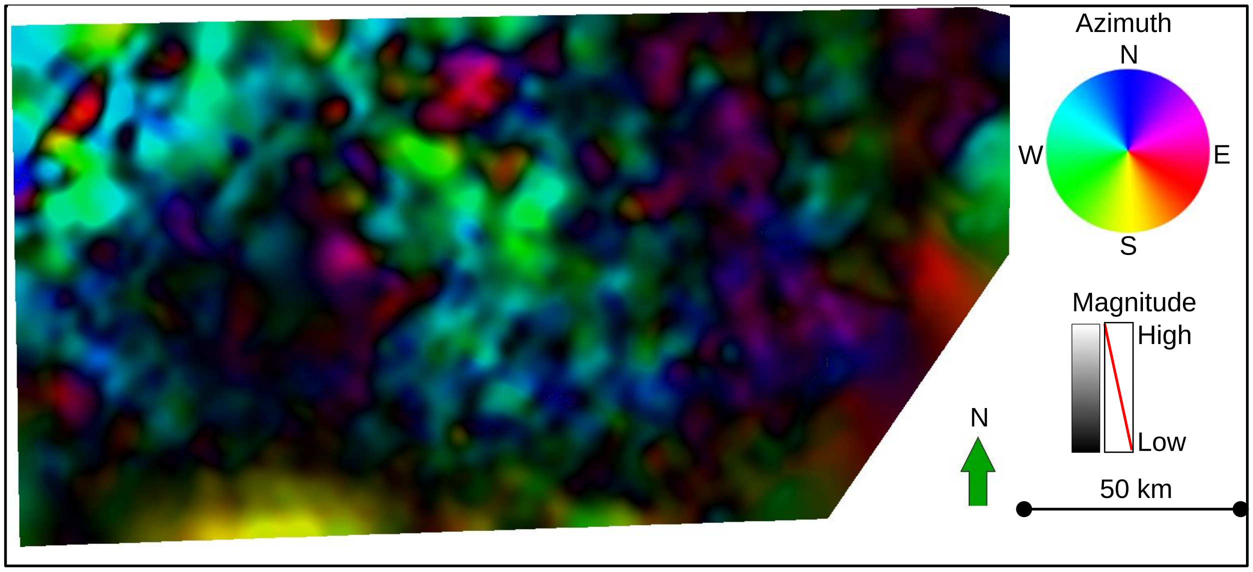


Figure 3.19. Co-rendered reflector convergence magnitude and azimuth extracted around the top of the Muderong Shale formation. Low magnitude is black and opaque, while high magnitude is transparent, allowing the colors of azimuth to be visible. Generally, the layers are thinning toward the NW (green and cyan color), with some exception in the middle and eastern part of the study area where they are thinning toward the NE (purple color).

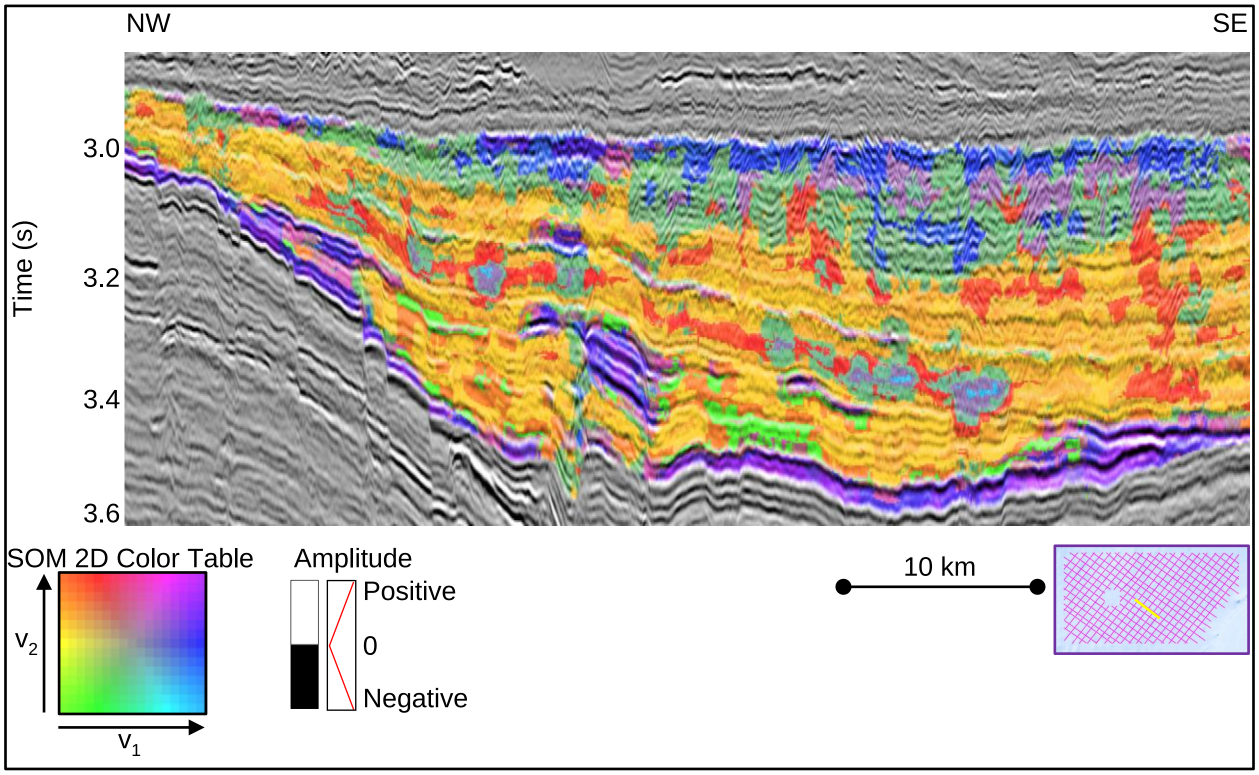
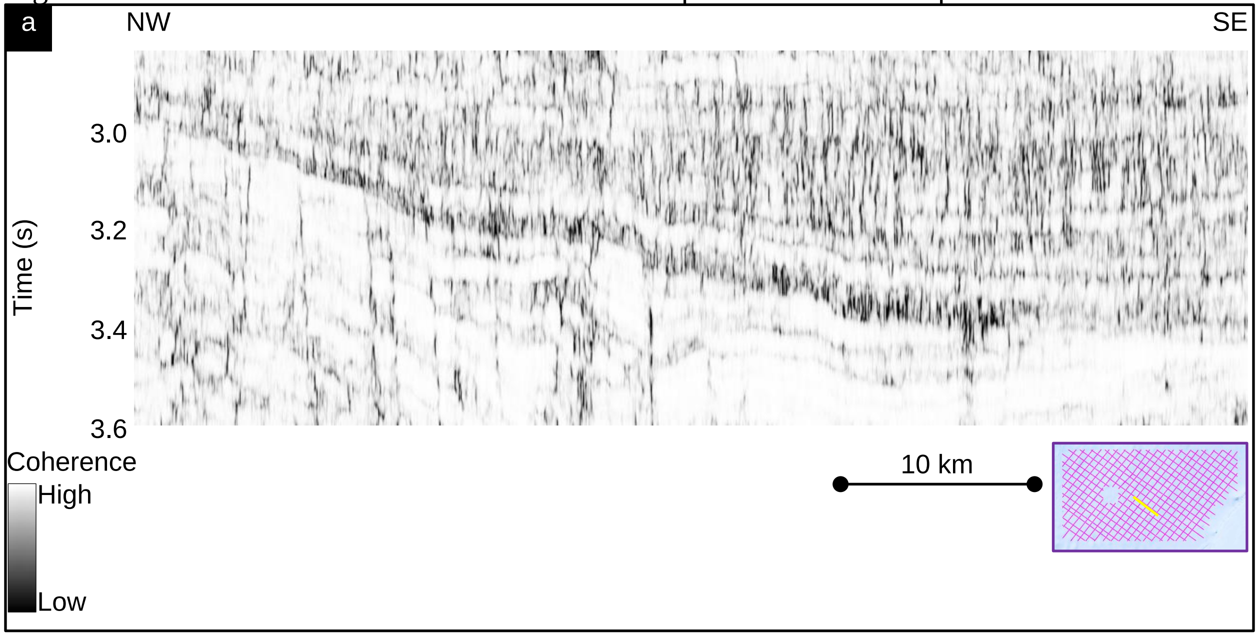


Figure 3.20. SOM result corendered with seismic amplitude on the same portion of a strike line in



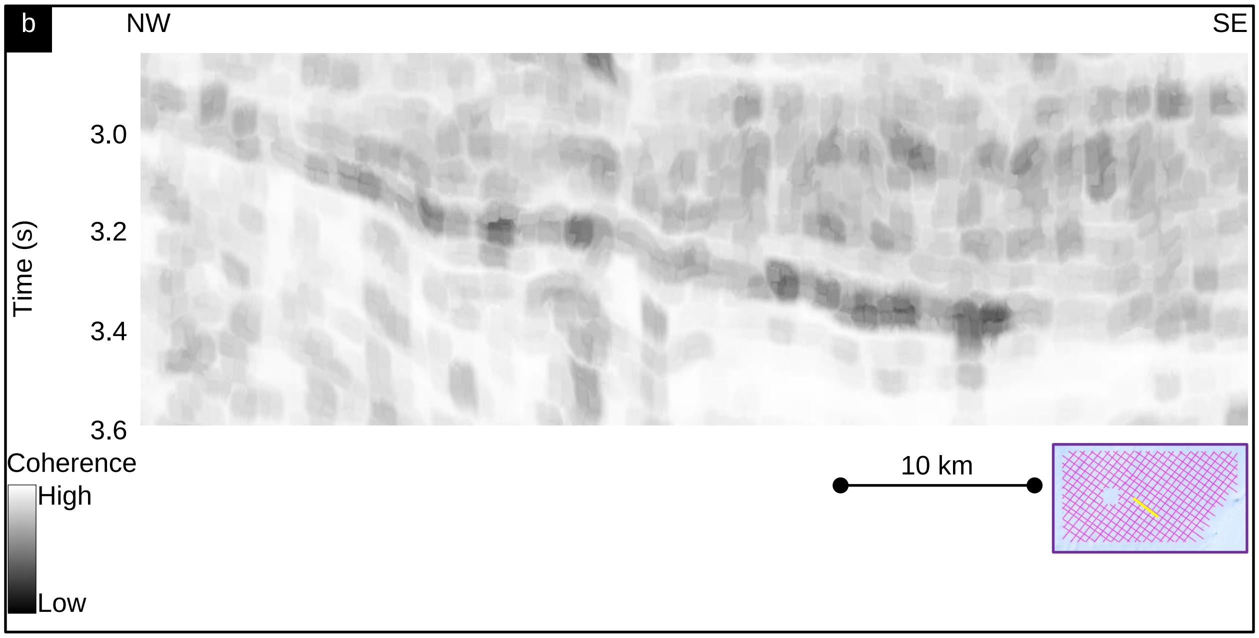


Figure 3.10. v_1 and v_2 are the first two eigenvectors of the four attributes used in SOM computation. SOM classification is performed only within the Muderong Shale. Red, light cyan, light purple, and blue colors correspond to chaotic portion of MTCs. Yellow and orange colors correspond to low amplitude, conformal reflectors. Bright green color corresponds to moderate amplitude, continuous reflectors. Bright purple and dark violet correspond to high amplitude, continuous reflectors.

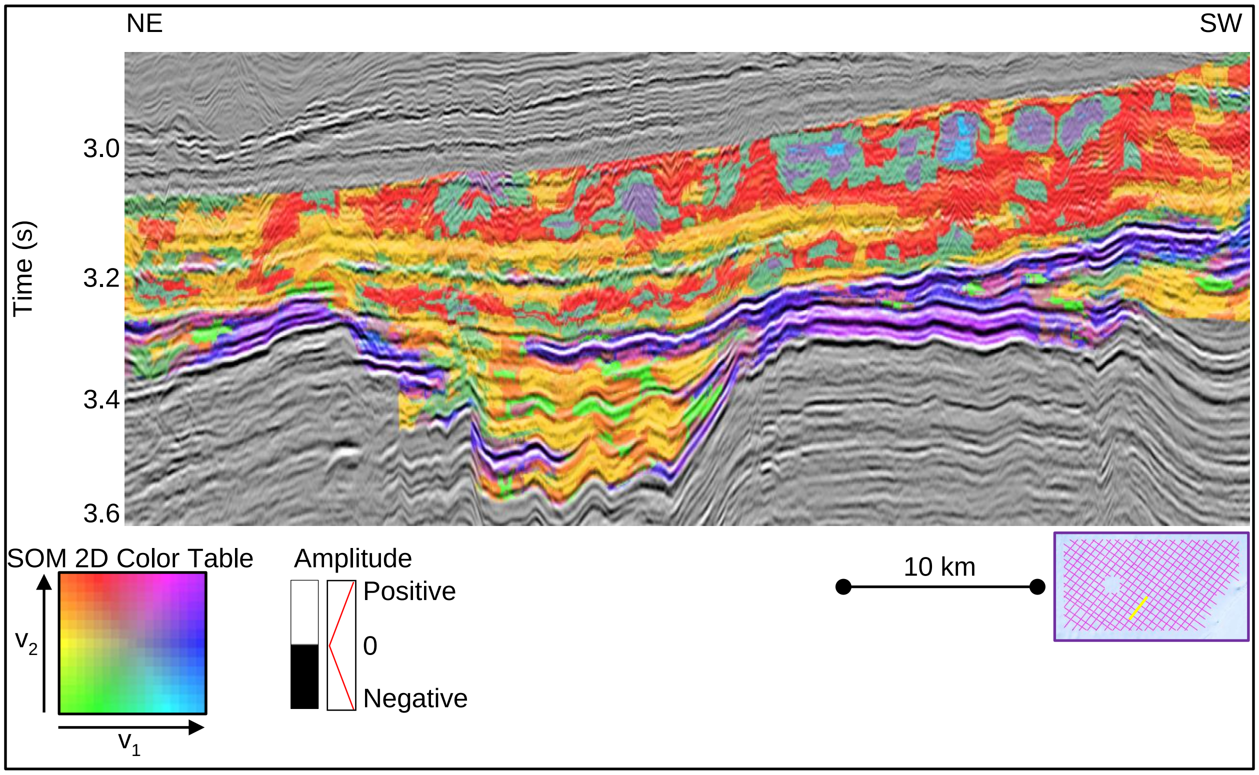


Figure 3.21. SOM result corendered with seismic amplitude on the same portion of a dip line in Figure 3.11. v_1 and v_2 are the first two eigenvectors of the four attributes used in SOM computation. SOM classification is performed only within the Muderong Shale. Red, light cyan, light purple, and blue colors correspond to chaotic portion of MTCs. Yellow and orange colors correspond to low amplitude, conformal reflectors. Bright green color corresponds to moderate amplitude, continuous reflectors. Bright purple and dark violet correspond to high amplitude, continuous reflectors.

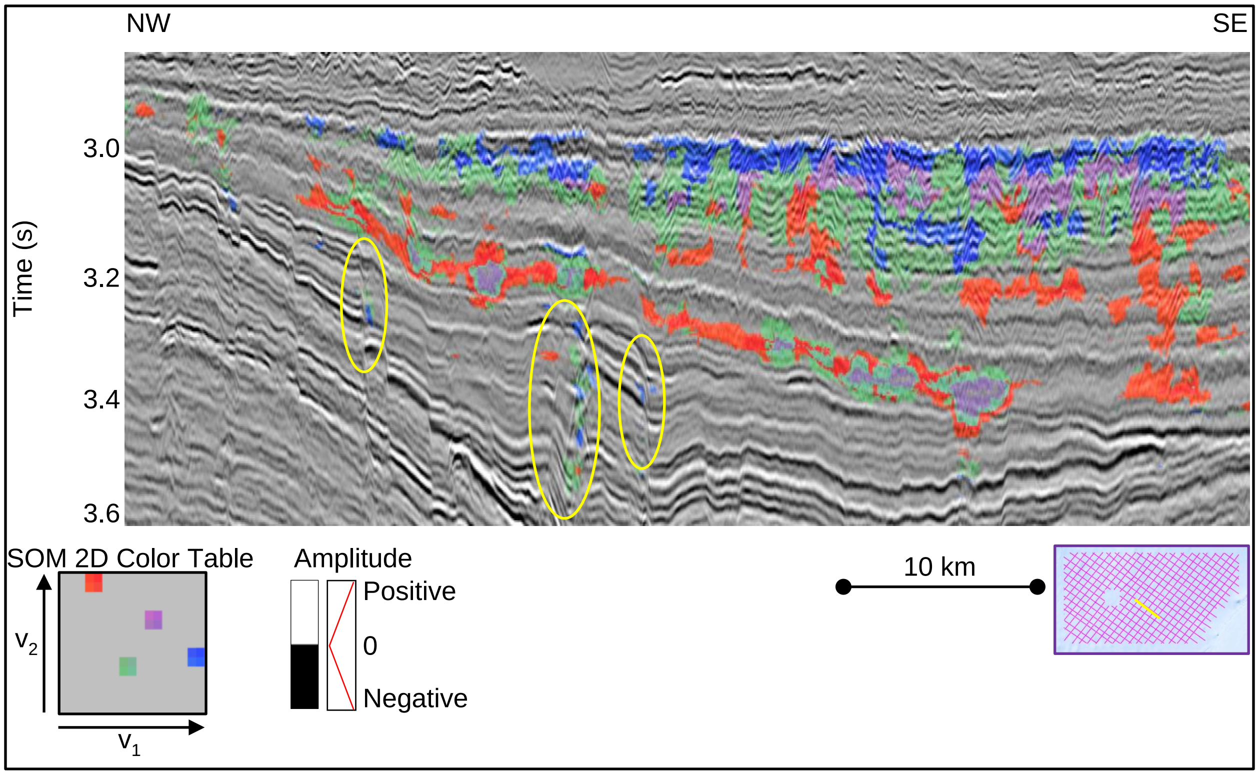


Figure 3.22. The same portion of a strike line in Figure 3.20, keeping only red, light cyan, light purple, and blue colors of SOM result. The rest of the SOM 2D color table is set to a neutral gray color. The result is a co-rendered image of seismic amplitude and MTC "clusters." Note that some bow-tie artifacts, which have the same chaotic expression as MTCs, are also marked by the colors of these MTC "clusters" (yellow ellipses).

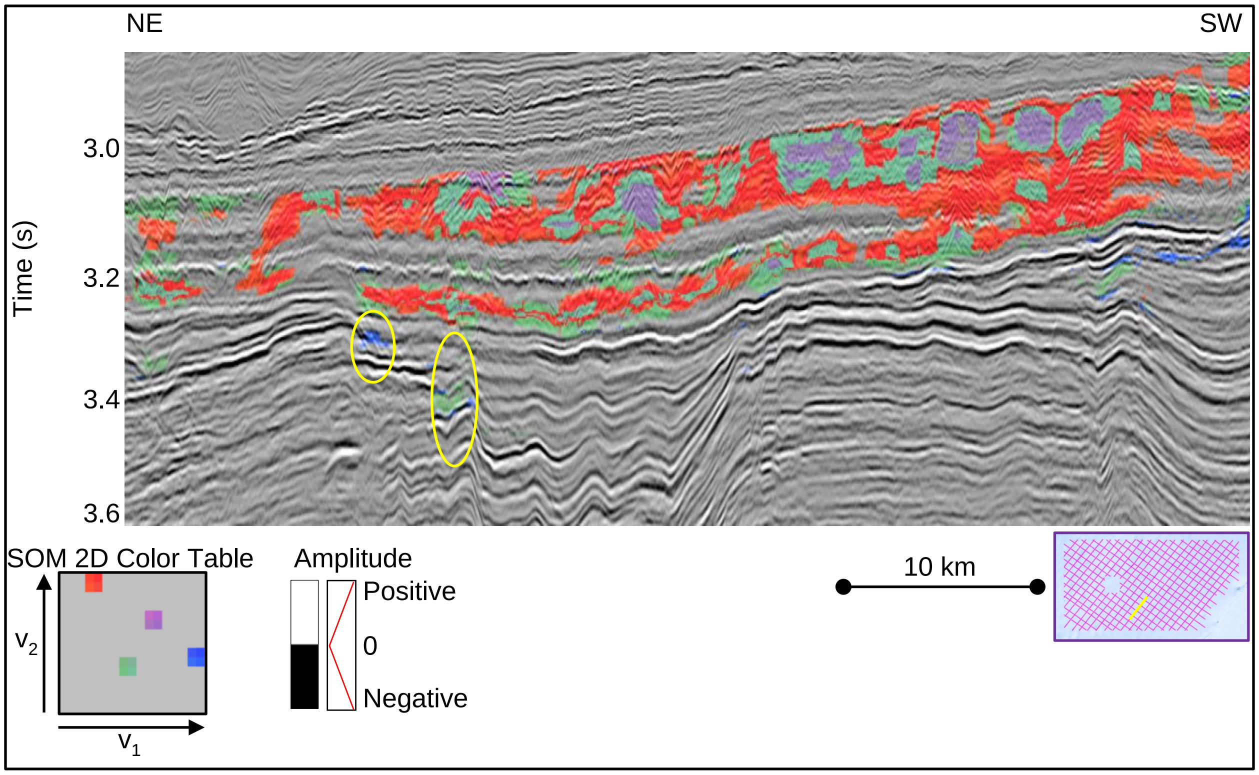


Figure 3.23. The same portion of a dip line in Figure 3.21, keeping only red, light cyan, light purple, and blue colors of SOM result. The rest of the SOM 2D color table is set to a neutral gray color. The result is a co-rendered image of seismic amplitude and MTC "clusters." Note that some bow-tie artifacts, which have the same chaotic expression as MTCs, are also marked by the colors of these MTC "clusters" (yellow ellipses).

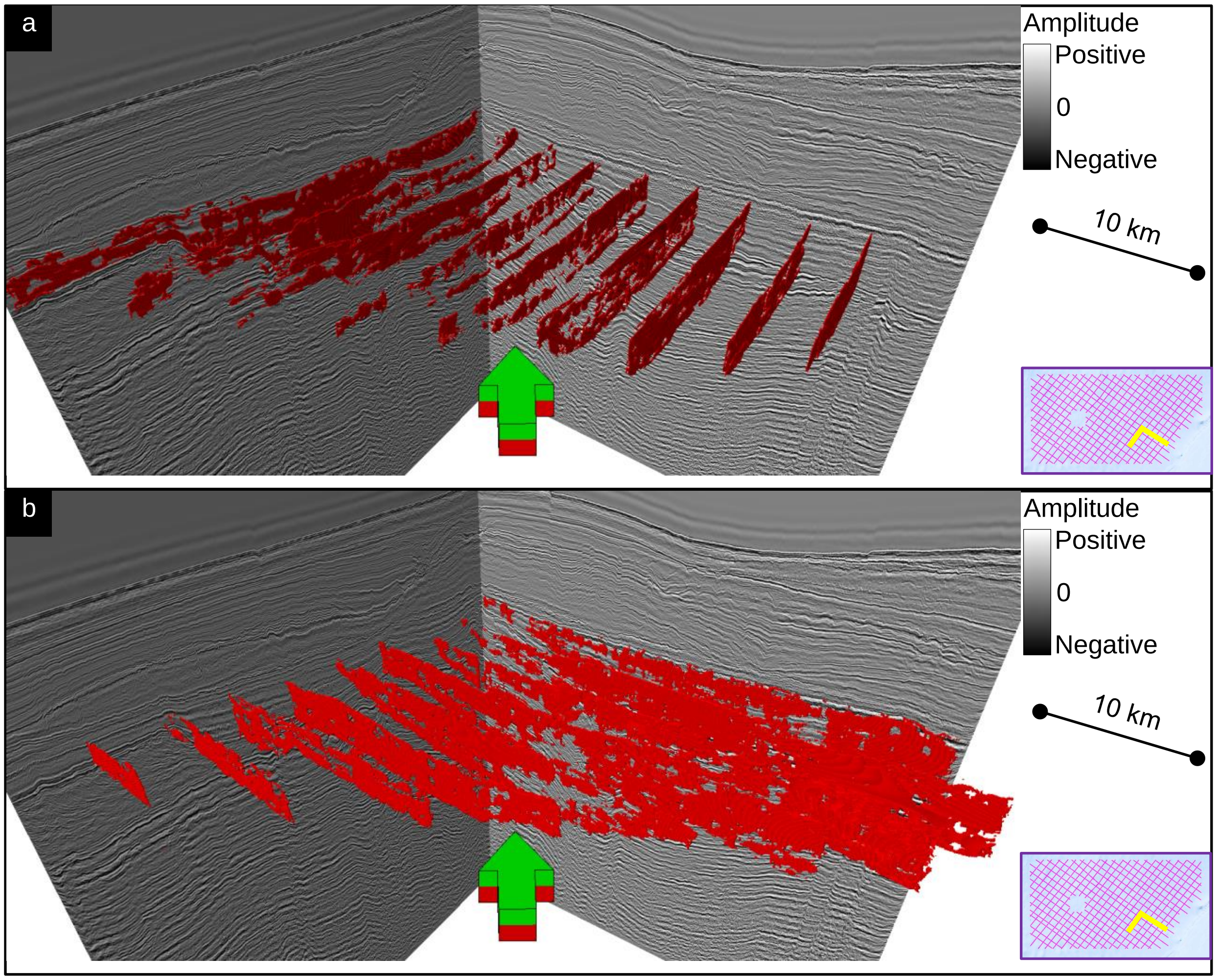


Figure 3.24. 3D perspective view showing MTC geobodies extracted from (a) dip lines and (b) strike lines. This gigantic MTC sequence is located at the SE part of the survey, in the bottom of the Tertiary carbonate sequences, extending more than 50 km in the dip direction and 80 km in the strike direction.

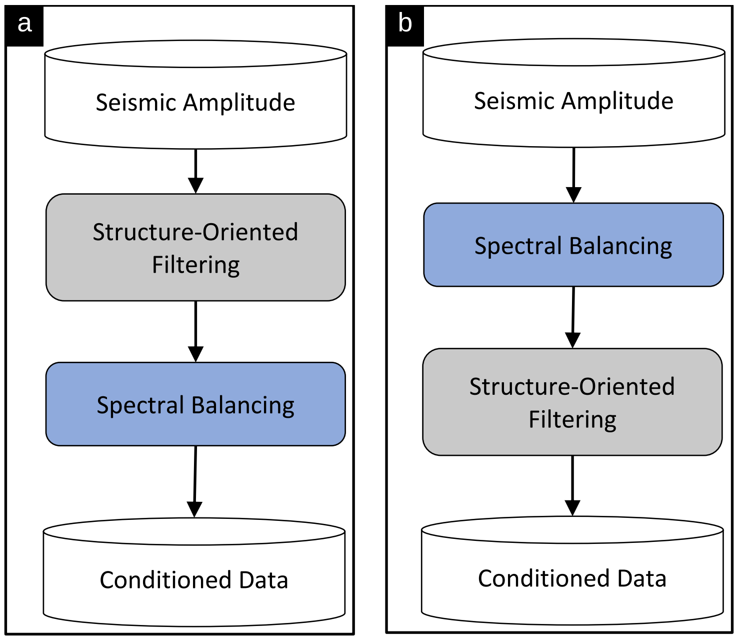


Figure 3.25. Different order of filters in data conditioning workflow. Should I apply (a) structure-oriented filtering first or (b) spectral balancing first?

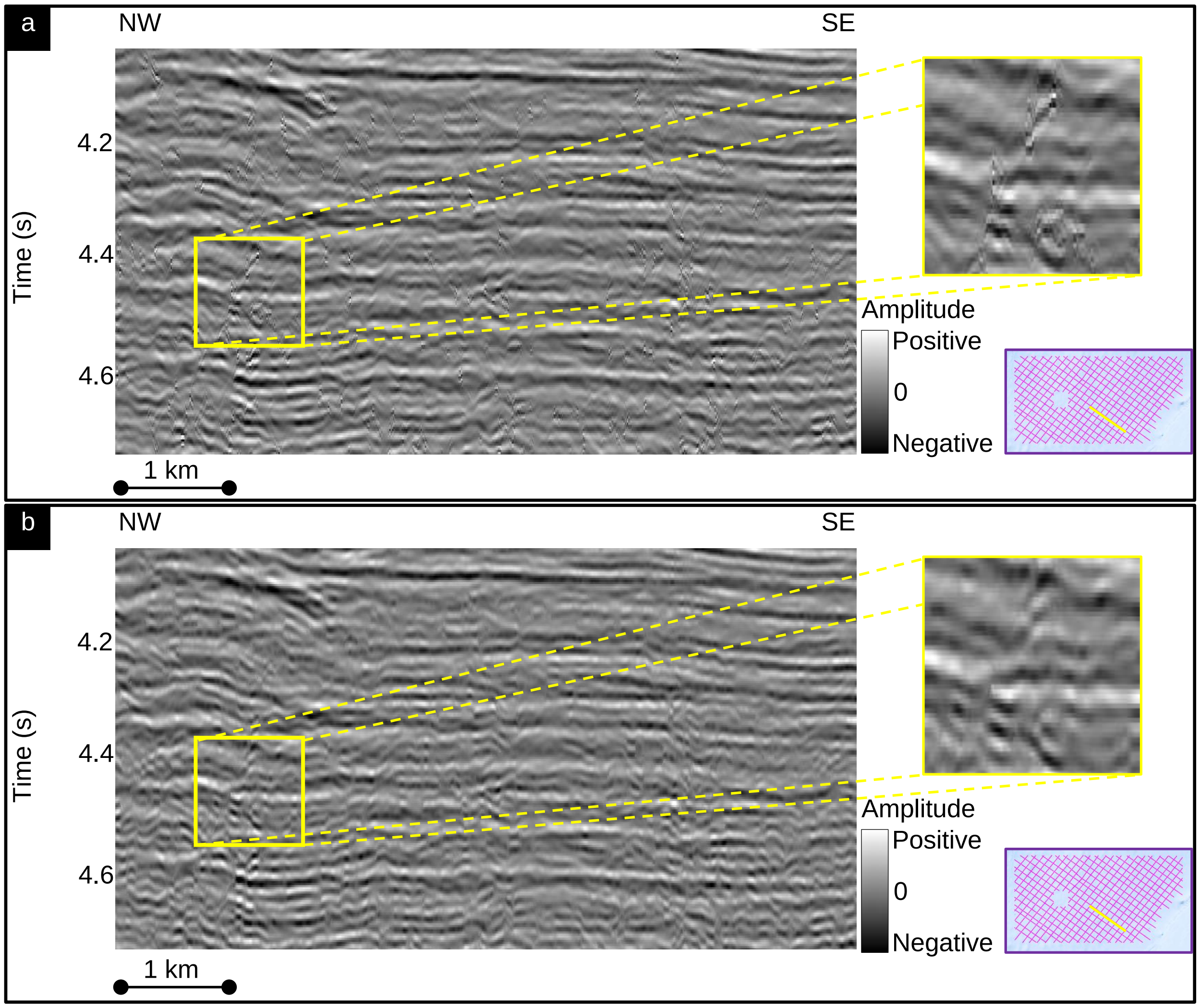


Figure 3.26. A section of a 2D line (a) with structure-oriented filtering applied first, followed by spectral balancing, and (b) with spectral balancing first, followed by structure-oriented filtering. Note the artifacts present in case (a) that are absent from case (b).

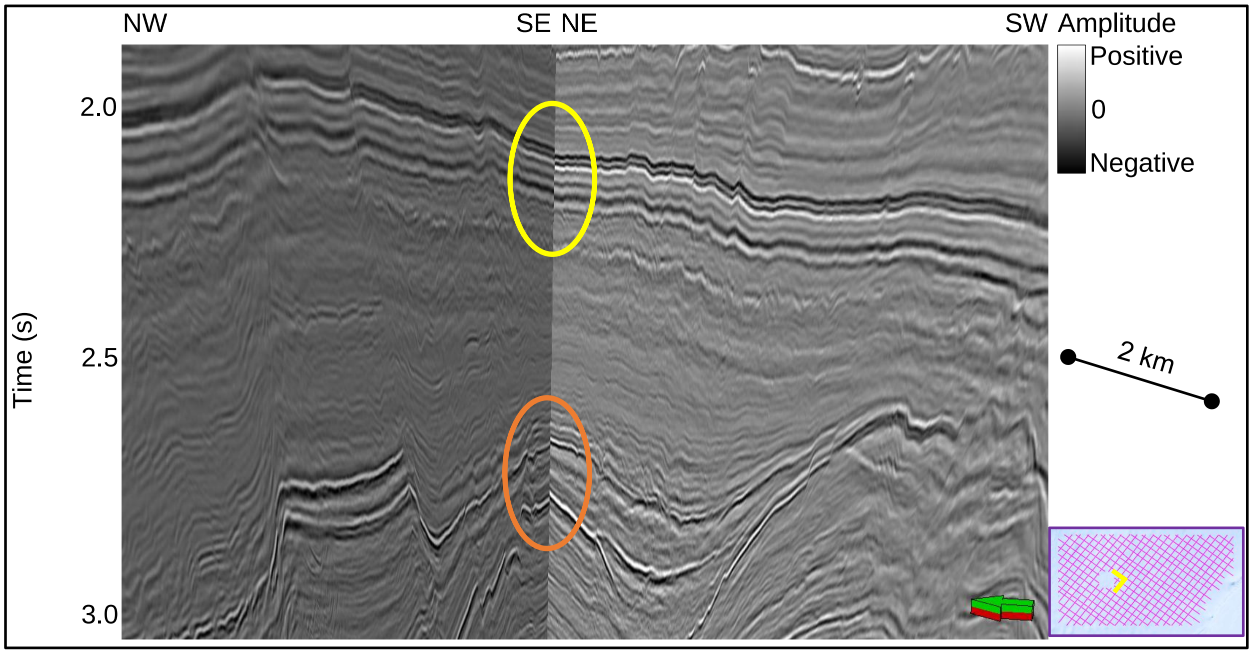


Figure 3.27. 3D perspective view showing reflector mismatches at the intersection of two perpendicular vertical seismic profiles near the center of the survey. Note the yellow ellipse, where reflectors on the right appear to arrive later than those on the left, and the orange ellipse, where reflectors on the right appear to arrive sooner than those on the left. These mismatches make it difficult to pick a consistent horizon across all lines.

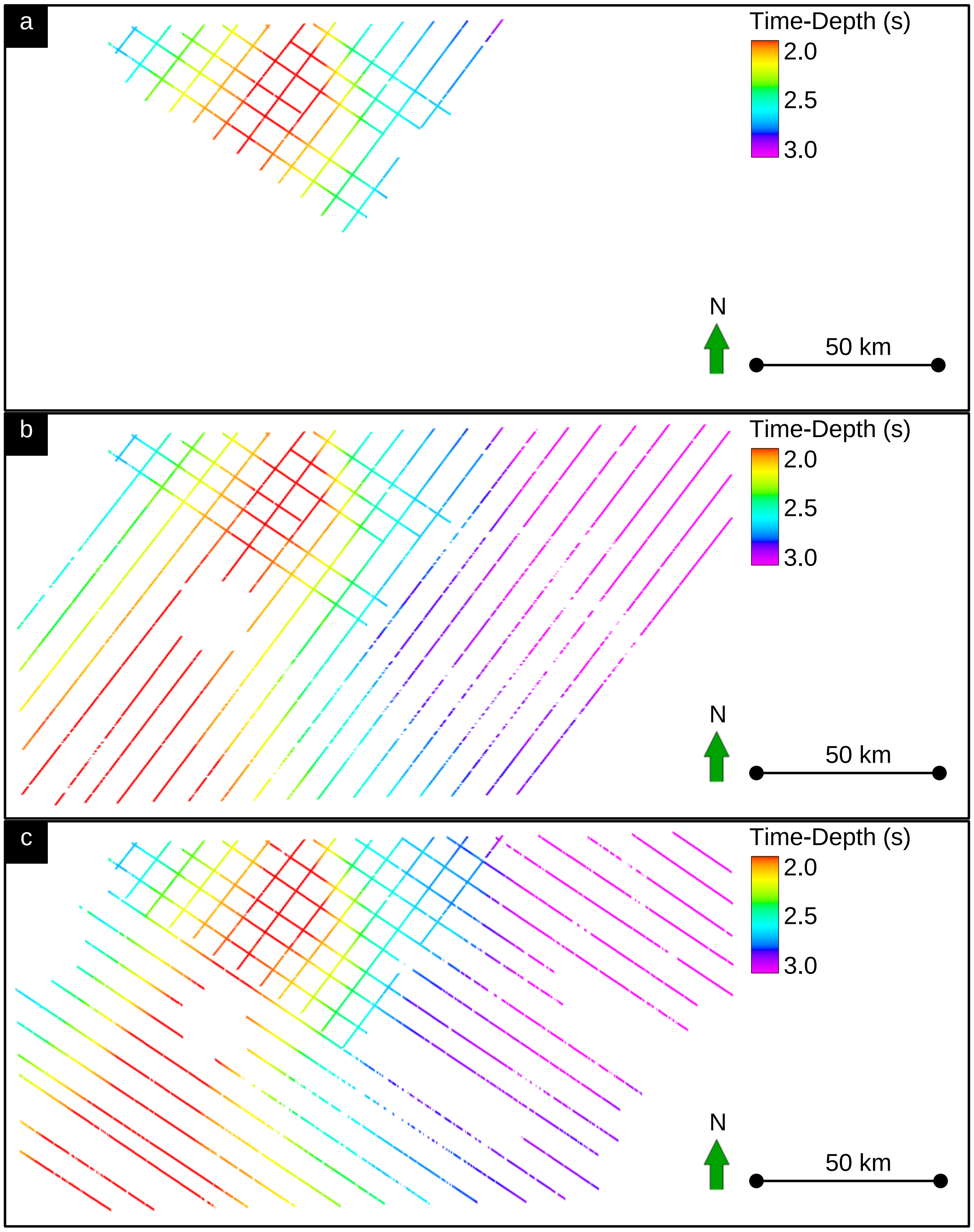


Figure 3.28. Workaround for reflector misties: first, pick horizon where reflectors tie (a). Then, make two copies of the common picks in (a) and continue to define two separate sets of horizon picks: one parallel to the lines trending NE-SW (b), and another one parallel to the lines trending NW-SE (c).

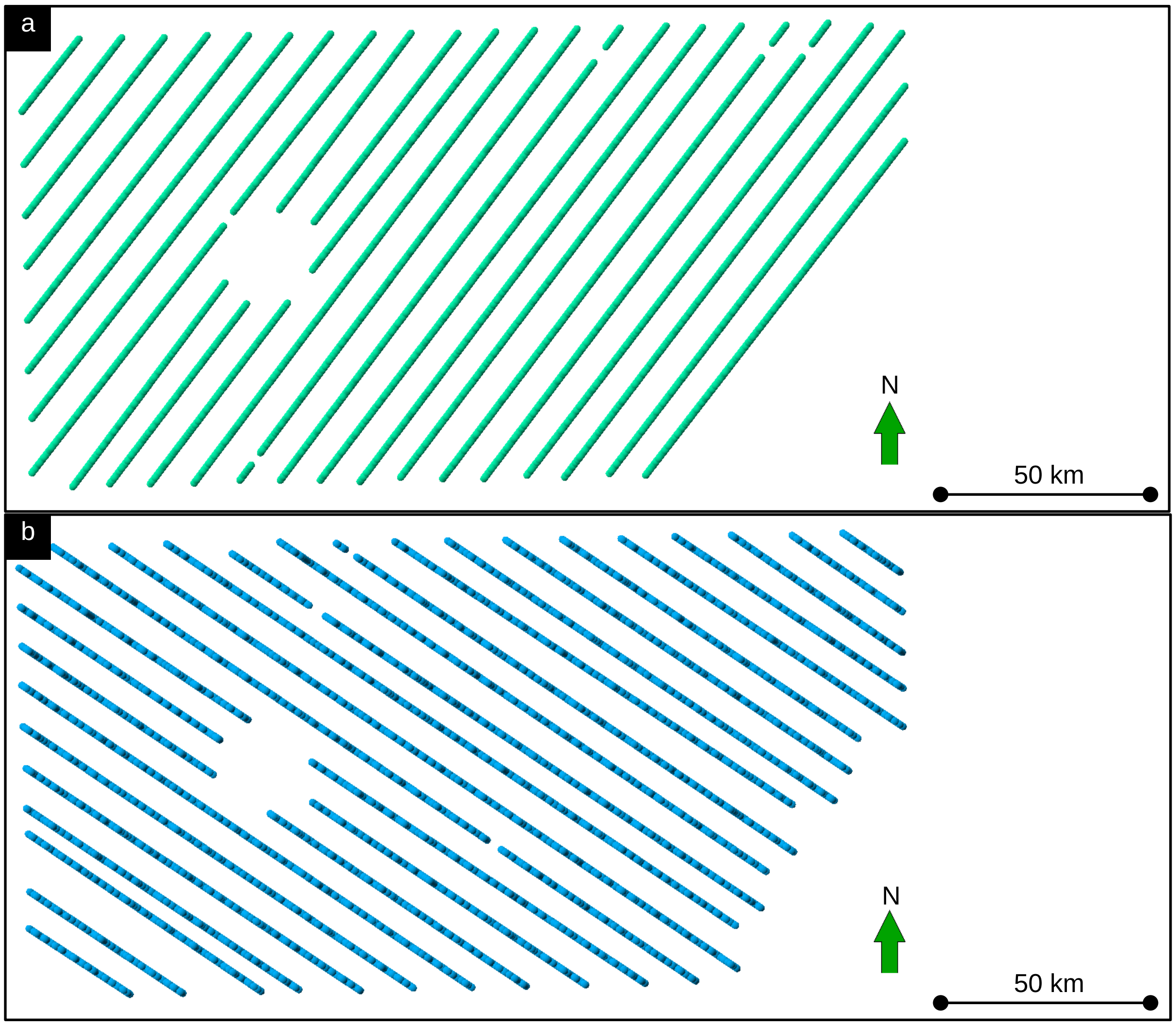


Figure 3.29. First step in reflector convergence extraction along a surface: perform attribute extraction on two separate sets of lines, one trending NE-SW (a), and one trending NW-SE (b). Make sure all lines in a set have the same direction of increasing CDP number.

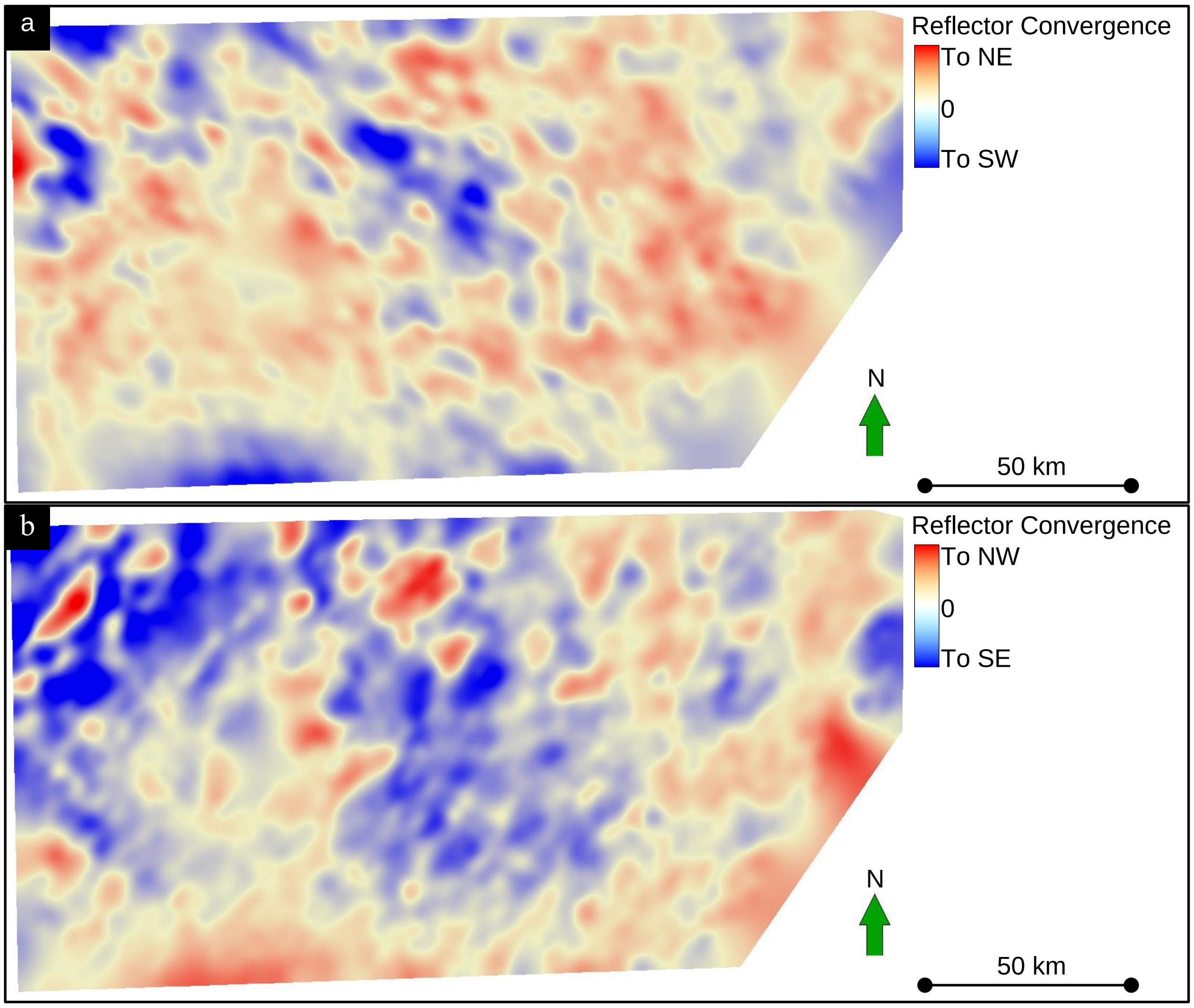


Figure 3.30. Second step in reflector convergence extraction along a surface: make two surfaces from the two separate extracted point sets: one for lines trending NE-SW (a), and one for lines trending NW-SE (b). These surfaces can be converted to reflector convergence's magnitude and azimuth (Figure 3.19) using equation (3.1) and (3.2).

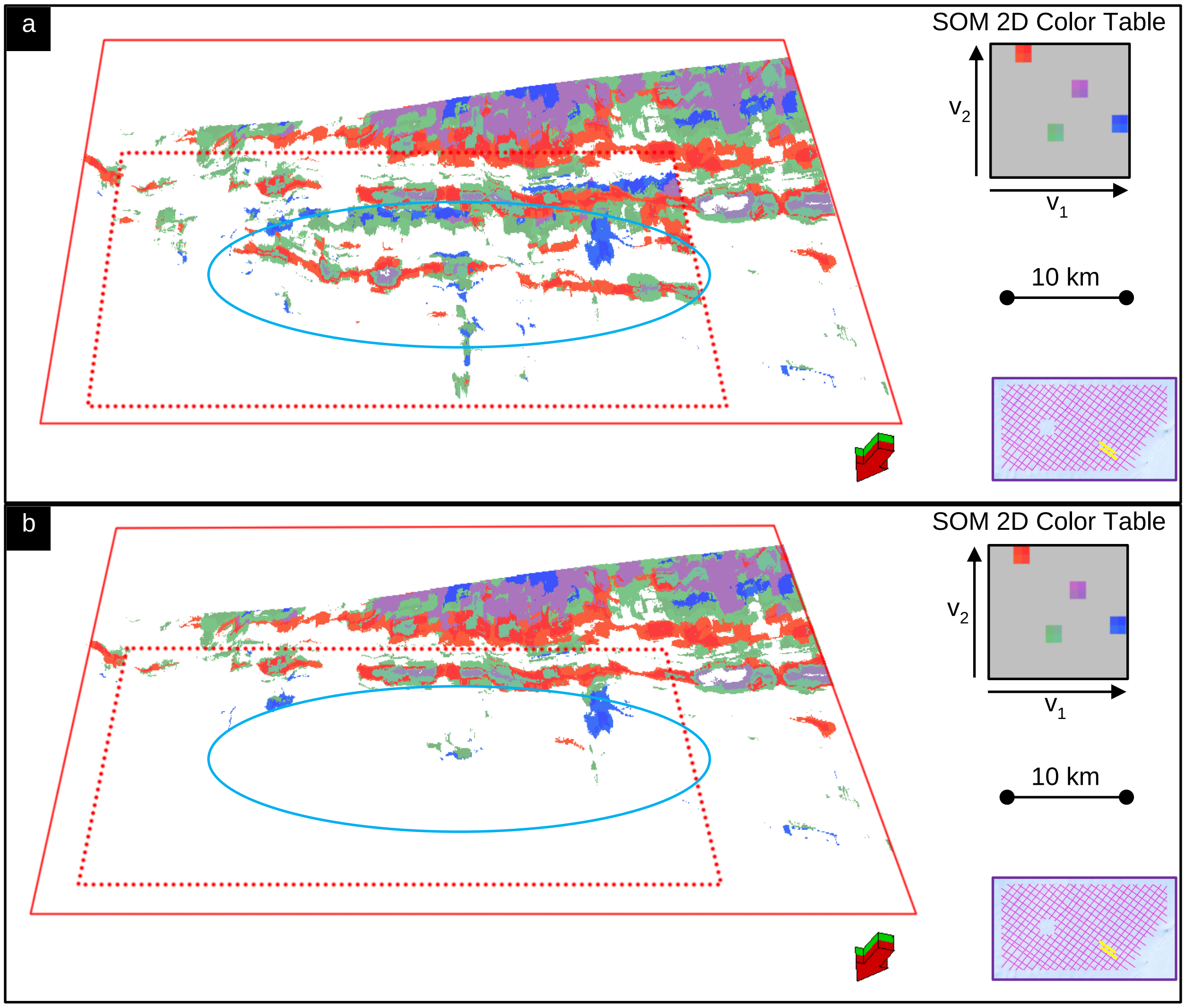


Figure 3.31. Transparency approach to display MTC clusters in 3D using a commercial interpretation software. The line farthest from the viewpoint is fully rendered at an angle of view (a) while not rendered correctly at a slightly different angle of view (b). MTC clusters in the cyan ellipse is rendered in (a) and not rendered in (b).

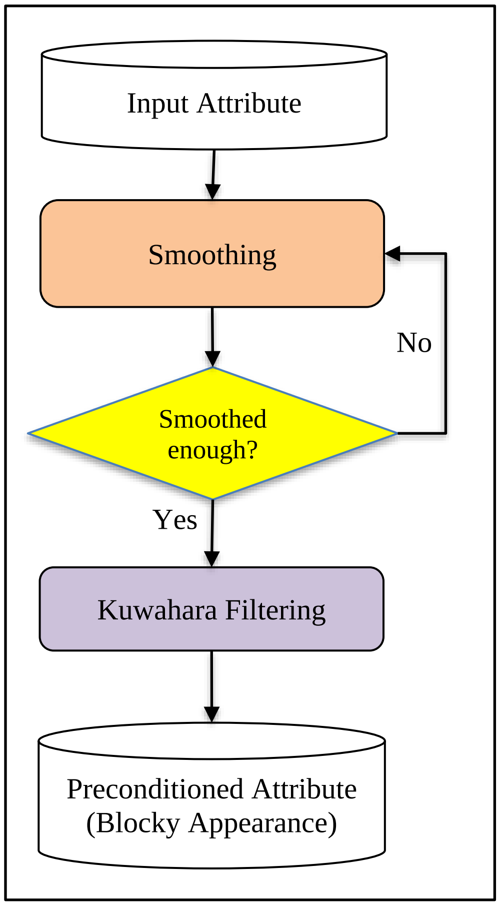


Figure 3.C-1. Attribute preconditioning workflow (modified from Qi et. al., 2016).

Tables

Table 3.1. Limitations of different attribute types regarding data dimensionality.

Attribute	1D Data	2D Data	3D Data
Instantaneous attributes	Envelope, frequency, phase,…	Envelope, frequency, phase,…	Envelope, frequency, phase,…
Spectral decomposition	Spectral components	Spectral components	Spectral components
Coherence	Not applicable	2D discontinuities	3D discontinuities
Dip	Not applicable	Apparent dip	Vector dip
Curvature	Not applicable	Apparent (Euler) curvature	k_1 and k_2 curvature and strike
Texture (GLCM)	Not applicable	2D textures	3D textures
Reflector convergence	Not applicable	Apparent convergence	Vector convergence

Table 3.2. Simplified stratigraphic column of the North Carnarvon Basin (based on geological description from Tellez Rodriguez, 2015).

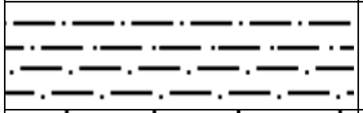
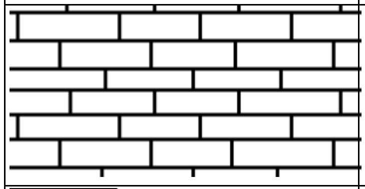
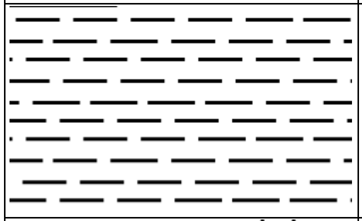
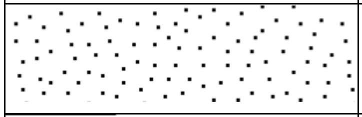
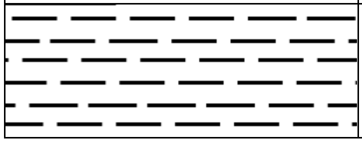
Lithology	**Description**
	Quaternary sediment
	Tertiary carbonate: Mandu limestone
	Middle Cretaceous transgressive sequences: Muderong Shale
	Middle Triassic - Early Cretaceous sequences: Mungaroo fluvio-deltaic formation, Barrow delta (Flag Sandstone)
	Early Triassic deposit: Locker shale

Table 3.3. Attribute expressions of seismic facies.

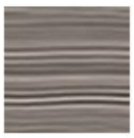
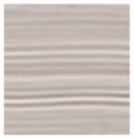
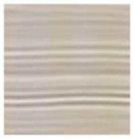
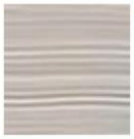
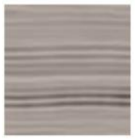
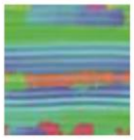
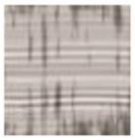
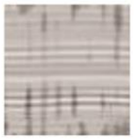
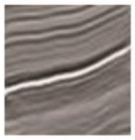
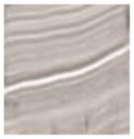
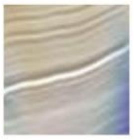
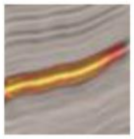
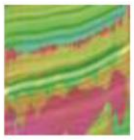
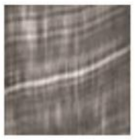
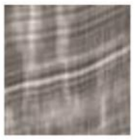
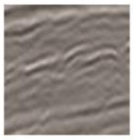
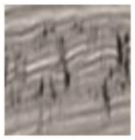
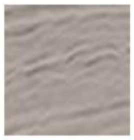
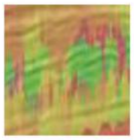
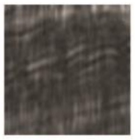
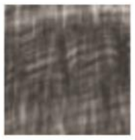
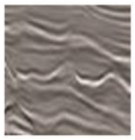
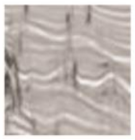
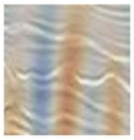
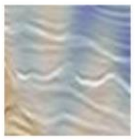
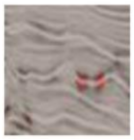
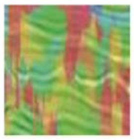
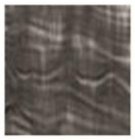
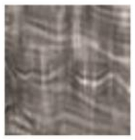
Facies	Amplitude	Coherence	Curvatures	Reflector Convergence	Coherent Energy	Peak Frequency	GLCM Energy	GLCM Homogeneity
Conformal Reflectors								
Anomalously Strong Reflectors								
MTCs								
Channels								

Table 3.A-1. Spectral decomposition parameters.

Parameter	**Value**
Average spectrum smoothing window half height (s)	0.5
Spectral balancing factor (%)	4
Bluing exponent	0
Line and CDP decimation	1
Ormsby filter corner point f_1 (Hz)	5
Ormsby filter corner point f_2 (Hz)	10
Ormsby filter corner point f_3 (Hz)	90
Ormsby filter corner point f_4 (Hz)	120
CWT mother wavelet bandwidth (Hz)	0.26051
Temporal taper (s)	0.1
Percentile excluded in spectral shape	0.15
Lowest output frequency f_{low} (Hz)	5
Highest output frequency f_{high} (Hz)	100
Output frequency increment Δf (Hz)	5

Table 3.A-2. Structural dip computation parameters.

Parameter	**Value**
Algorithm	Semblance Search
Maximum angle searched (degree)	20
Search angle increment (degree)	5
Time-to-depth conversion velocity (m/s)	4000
Vertical window half height (s)	0.02
Inline window radius (m)	31.25
Crossline window radius (m)	0

Table 3.A-3. Dip filtering parameters.

Parameter	**Value**
Algorithm	LUM
Lower-Upper-Median (LUM) percentile	20
Vertical window half height (s)	0.02
Inline window radius (m)	31.25
Crossline window radius (m)	0

Table 3.A-4. Similarity computation parameters.

Parameter	**Value**
Vertical window half height (s)	0.02
Inline window radius (m)	31.25
Crossline window radius (m)	0
Low cut filter rolloff $\boldsymbol{f}_{\mathbf{low}}$ (Hz)	5
High cut filter rolloff $f_{\mathbf{high}}$ (Hz)	100

Table 3.A-5. Structure-oriented filtering parameters.

Parameter	**Value**
Vertical window half height (s)	0.02
Inline window radius (m)	31.25
Crossline window radius (m)	0
Similarity value $\boldsymbol{s}_{\mathbf{low}}$, below which 0% of the filter is applied	0.3
Similarity value $\boldsymbol{s}_{\boldsymbol{high}}$, above which 100% of filter is applied	0.4
Similarity value $\boldsymbol{s}_{\mathbf{center}}$, above which smoothing takes place in a centered (rather than in the best Kuwahara) window	0.4
Algorithm	Principle-Component

Table 3.B-1. Curvature and reflector convergence computation parameters. The four values of (λ_j,w_j) define a long-wavelength filter in the wavelength domain applied to the first derivative operators in the inline, crossline, and vertical directions applied to the input inline and crossline dip components. The time-to-depth conversion velocity is the same as used to compute dip.

Parameter	**Value**
Curvature type	Structural
Filter corner point λ_1 (m)	200,000
Filter corner point λ_2 (m)	2,000
Filter corner point λ_3 (m)	1,000
Filter corner point λ_4 (m)	500
Filter weight w_1	1
Filter weight w_2	0.666
Filter weight w_3	0.333
Filter weight w_4	0

Table 3.B-2. GLCM attributes (energy and homogeneity) computation parameters.

Parameter	**Value**
Vertical window half height (s)	0.02
Inline window radius (m)	31.25
Crossline window radius (m)	0
Number of gray levels	33

Table 3.C-1. Smoothing parameters.

Parameter	Value
Number of iterations	2
Vertical window half height (s)	0.012
Inline window radius (m)	312.5
Crossline window radius (m)	0

Table 3.C-2. Kuwahara filtering parameters.

Parameter	Value
Vertical window taper (%)	20
Vertical window half height (s)	0.012
Inline window radius (m)	125
Crossline window radius (m)	0

Table 3.C-3. SOM classification parameters, where the decimation factors define the data used to train the SOM. Most interpretation software packages are limited to 256 colors, thereby (minimally) constraining the maximum number of potential classes.

Parameter	Value
Number of prototype vectors (maximum number of classes)	256
Number of standard deviations along the first two eigenvector directions to define the initial 16*16=256 prototype vectors on the manifold	±4
Initial Neighborhood Scale	1.2
Distance type	z-scored Euclidian
Maximum number of training iterations	50
CDP decimation	5
Line decimation	1
Vertical sample decimation	1
Grid spacing	150
Upper horizon	Top Muderong Shale
Lower horizon	Top Triassic

CHAPTER 4: AN IN-DEPTH ANALYSIS OF LOGARITHMIC TRANSFORMATION AND PER-CLASS NORMALIZATION IN MACHINE LEARNING: APPLICATION TO UNSUPERVISED CLASSIFICATION OF A TURBIDITE SYSTEM IN THE CANTERBURY BASIN, NEW ZEALAND AND SUPERVISED CLASSIFICATION OF SALT IN THE EUGENE ISLAND MINI-BASIN, GULF OF MEXICO

Abstract

In a machine learning workflow, data normalization is a crucial step that compensates for the large variation in data ranges and averages associated with different types of input measured with different units. However, most machine learning implementations do not provide data normalization beyond the z-score algorithm which subtracts the mean from the distribution and then scales the result by dividing by the standard deviation. Although z-score converts data with Gaussian behavior to have the same shape and size, the majority of seismic attributes exhibit log-normal, or even more complicated distributions. Because many machine learning applications are based on Gaussian statistics, I wish to evaluate the impact of more sophisticated data normalization techniques on the resulting classification. To do so, I provide an in-depth analysis of data normalization in machine-learning classifications by formulating and applying a logarithmic data transformation scheme to the unsupervised classifications (including PCA, ICA, SOM, and GTM) of a turbidite channel system in the Canterbury Basin, New Zealand, as well as implementing a per-class normalization scheme to the supervised probabilistic neural network (PNN) classification of salt in the Eugene Island mini-basin, Gulf of Mexico. Compared to the simple z-score normalization, a single logarithmic transformation applied to each input attribute significantly increases the spread of the resulting clusters (and corresponding color contrast), thereby enhancing subtle details in projection and unsupervised classification. However, this

same uniform transformation produces less-confident results in supervised classification using probabilistic neural networks. I find that more accurate supervised classifications can be obtained by applying class-dependent normalization for each input attribute.

Introduction

Machine learning has been applied to seismic facies classification for more than 20 years, with early successes reported for both supervised classification by Meldahl et al’s (1999) and West et al. (2002) and for unsupervised classification by Poupon et al. (1999) and Strecker and Uden (2002). Since that time, a wide variety of both commercial and research implementation of machine learning facies classification algorithms have been adopted by the seismic interpretation community. A key assumption of most machine learning workflows is that the input data approximate Gaussian-shaped distributions. However, if I wish to measure the distance of a given data point to a cluster center, I need to account for the differences in units. For example, P-wave impedance may exhibit values that range from 1,500 to 12,000 g/cm^3.m/s, whereas Poisson’s ratio may vary from 0.1 to 0.45. In this case, a simple z-score normalization of each sample *j* of the *k*th attribute, a_{jk}, by its mean, μ_k, and standard deviation, σ_k,

$$\bar{a}_{jk} = \frac{a_{jk}-\mu_k}{\sigma_k} \tag{4.1}$$

removes the effect of using different measurement units and assigns equal importance to each measurement. Although seismic amplitude and some seismic attributes can be well represented by a Gaussian distribution, most seismic attributes are skewed, while some may exhibit an approximately uniform distribution (e.g. cosine of instantaneous phase), suggesting the need for a nonlinear transformation (Figure 4.1).

Generally, machine-learning classifications can be categorized into two major types: (1) unsupervised classification, which automatically assigns different colors to different groups of data, and (2) supervised classification, which requires a human interpreter to explicitly label a subset of the data as belonging to a given class or seismic facies.

Unsupervised Classifications

Unsupervised classifications, in turn, can be divided into two categories: (1) projection techniques, in which *N* input attributes are projected onto a latent space of smaller dimensionality (usually 2-3 dimensions allowing mapping against a 2D or 3D continuous color matrix), and (2) clustering techniques, in which different data points are arranged into different clusters, with each cluster assigned with a distinct color.

The most well-known projection technique is Principal Component Analysis (PCA), in which the input attributes are projected onto two or more eigen vectors of their covariance matrix in order to capture the maximum variation of the input data (Guo et al., 2008; Chopra and Marfurt, 2014). Although the principal components are orthogonal and theoretically uncorrelated, seismic facies are not, such that PCAs can mix different seismic facies. Honorio et al. (2014) and Lubo-Robles and Marfurt (2019) address this issue using Independent Component Analysis (ICA), which is a non-orthogonal projection method based on higher order statistics. Among the latest projection methods is a stochastic, non-linear projection technique described by Wallet and Ha (2019), which utilize an autoencoder deep-learning neural network to "encode" *N* input attributes into a 3-dimensional latent space to be displayed via RGB blending.

Zhao et al. (2015) applied different clustering techniques, including K-means, Self-Organizing Maps (SOM), and Generative Topographic Mapping (GTM) to a turbidite system in the Canterbury Basin, New Zealand. Among those clustering techniques, K-means is the simplest and fastest method, in which the input training data are "partitioned" into a user-defined number of clusters based on the distance from a data point to the clusters' centers. In general, K-means does not take into account the size and shape of each cluster in the classification. In contrast, an extension of K-means results in a Gaussian Mixture Model (GMM), in which each cluster is assigned its own multidimensional Gaussian density functions. The size, shape, and position of each Gaussian density function is modified in an iterative expectation-maximization scheme (Hardisty and Wallet, 2017). A common drawback of both K-means and GMM algorithms is that the classification result does not graphically show the proximity of one cluster to another. Instead, SOM directly maps the clusters to a deformed manifold, such that the relative position of each cluster is defined (Zhao et al., 2015). The manifold is in turn mapped to a latent space amenable to color mapping. While early SOM algorithms (e.g. Poupon et al., 1999) used a 1D curve mapped against a rainbow color bar as the manifold, 2D deformed surfaces mapped against a 2D color bar are now more common (e.g. Strecker et al., 2005; Castro de Matos et al., 2010; Roden et al., 2015; Zhao et al., 2015), although 3D manifolds can be mapped against RGB. Castro de Matos et al. (2010) and Zhao et al. (2016) used Sammon mapping to better measure the distances in n-D attribute space to a deformed 2D manifold. Zhao and Marfurt (2017) evaluate different training data extraction schemes, constraining SOM analysis with stratigraphy (Zhao et al., 2017), and, most recently, applying a data-adaptive weighting scheme for SOM input attribute selection (Zhao et al., 2018). The original SOM algorithms perform clustering by finding the closest prototype vector (or neuron) to a data sample and does not provide a measure of the confidence of the clustering process (Roy et al., 2014; Chopra and Marfurt, 2014). More recent innovations construct a Gaussian distribution after clustering, which then provides a measure of the confidence of each clustered data point (Roden and Chen, 2017). In contrast, Generative Topographic Mapping (GTM) provides a direct measure of the likelihood that a data point falls within a given class by estimating the contributions of all the latent space grid points to a data sample using a mixture of Gaussian density functions (Zhao et al., 2015).

Supervised Classifications

Supervised classifications also consist of two main types: (1) semi-supervised classifications, in which a clustering algorithm is trained (or labeled) by an interpreter to associate a cluster with a specific class or seismic facies, and (2) purely-supervised neural network classifications, in which an input sample is fed through a number of hidden neuron layers to determine to which class, or “label”, it belongs.

Qi et al. (2016) show how the otherwise unsupervised GTM can be modified to be a semi-supervised classification algorithm. First, they apply a Kuwahara median filter to the input attributes to smoothen the interior of seismic facies, while sharpening the boundaries between different facies. The authors then define training samples belonging to N classes, which are mapped onto to a regularly gridded GTM “manifold” and assigned with the corresponding N Gaussian density functions, allowing the classification to generate a probability density function for each facies.

Many of the explicitly supervised neural network algorithms were applied to predicting the well log response from one or more seismic attribute volumes (Verma et al., 2012; Torres et al., 2018). Neural network application to well logs occurred first because the limited computational power of interpretation workstations was able to train the network for a suite of 1D well logs, but not on seismic data volumes that are typically two to three orders of magnitude larger in size. Meldahl et al. (1999) were among the first to classify seismic facies using artificial neural networks, starting with defining a single target facies (a gas chimney) and everything else. Here, the training data were attribute vectors selected by a skilled interpreter on the computer screen. West et al. (2002) developed an ANN algorithm that classified some seven different seismic facies. Since then, other machine learning algorithms have been evaluated. Zhao et al.

(2014) performed lithofacies classification in the Barnett Shale by applying Proximal Support Vector Machine (PSVM) on density, gamma ray, and sonic logs. Qi et al. (2019) used mud logs and applied this same algorithm to predict areas of low and high rate of penetration in drilling horizontal wells in the Mississippian cherty-lime reservoirs of Oklahoma. Verma et al. (2012) mapped high-frackability and high-TOC zones in the Barnett Shale by predicting a gamma ray volume from seismic attributes extracted along well paths, using Probabilistic Neural Network (PNN). Similarly, Torres et al. (2018) applied PNN on a suite of seismic attributes and inversion results extracted along well paths to map TOC distribution in the Woodford Shale, Oklahoma. With the rapid evolution of PC hardware, Lubo-Robles et al. (2021) were able to implement PNN in supervised seismic facies classification using 3D seismic attributes. Kim et al. (2019) also used 3D seismic attributes in their supervised classification based on random forest (RF) algorithm. The authors carefully selected relevant input attributes for their RF supervised classification by computing both linear and non-linear correlation coefficients between each pair of input attributes and rejecting redundant attributes that are highly correlated to the others.

Negligence to Data Normalization

The vast majority of machine learning publications, unsupervised and supervised alike, do not describe how the training data were normalized. Although some papers specify a z-score normalization step in their machine learning workflow, most do not mention normalization at all. A rare exception is a paper by Qi et al. (2020), outlining an attribute scaling step in an automatic attribute selection workflow for a supervised classification scheme based on GTM and GMM algorithms. The authors find that most of the input attributes used in their supervised classification, such as coherence, energy, and GLCM texture attributes, requires a logarithmic scaling scheme to better approximate the data distributions by Gaussian curves. However, the authors did not specify the reason behind their simple formula of logarithmic scaling, such as log(1.0-x) for coherence and log(5.0-x) for GLCM Entropy attribute. Neither did the authors provide a comparison between logarithmically scaled and z-score normalized final classification results.

Without a comprehensive analysis and formulation of data normalization in the literature, the goal of this research is to define a suite of useful normalization schemes using only basic elements of mathematics. To simplify my demonstration, I separate the mathematical formulation and pseudocode of my logarithmic transformation in Appendices A and B for seismic interpreters and computer scientists who want to incorporate my logarithmic transformation into their own workflows. I then implement and compare my logarithmic transformation and per-class normalization methods to the traditional z-score normalization and bulk normalization schemes in two case studies: (1) unsupervised classifications of a turbidite system in the Canterbury Basin, New Zealand, and (2) supervised classification of salt facies in the Eugene Island mini-basin, Gulf of Mexico. Finally, I analyze the histograms of the input data and the classification results to explain the differences made by using different normalization schemes.

Case Study 1: Unsupervised Classification of a Turbidite Channel System in The Canterbury Basin, Offshore New Zealand

Geologic Settings

The Canterbury Basin, New Zealand, is well known for the complexity of its turbidite channel systems. The study area is located in the northern Waka3D seismic survey, at the transition between continental shelf and slope (Figure 4.2). The Canterbury Basin underwent three main stages of geological deformation. A thick layer of clastic and coaly sediments was deposited during Middle to Late Cretaceous rifting and subsidence (Sutherland and Browne, 2003). Then, organic-rich black shale and widespread limestone were formed when the basin entered a transgression from the Late Cretaceous to Middle Tertiary (Cozens, 2011). Since the Late Tertiary, the basin underwent a regression due to uplifting and minor tectonic inversion, thus shifting the study area into a transition zone between continental shelf and slope, where many paleo-canyons and turbidite channels were formed (Zhao et al., 2016). These canyons and channels were filled with late Tertiary carbonate debris and thus are potential reservoirs for hydrocarbons (Wallet and Ha, 2019).

Methods

Figure 4.3 represents my unsupervised classification workflow. The first step is to select the input attributes that are relevant to the classification of turbidite facies. Following the work by Wallet and Ha (2019), I choose six attributes as the input for our unsupervised classification (Figure 4.4):

- (1) Coherent energy: for measuring reflector strength,
- (2) Structural Curvedness: for highlighting channel axes and levees,

- (3) GLCM entropy and (4) GLCM homogeneity (texture attributes): for differentiating architectural elements that have different seismic texture appearance,
- (5) Peak frequency and (6) Peak magnitude (spectral decomposition attributes): for measuring most dominant layer thicknesses and reflector strength.

To ensure the training data adequately represents different facies in a turbidite system, I follow the work by Zhao and Marfurt (2017) and Zhao et al. (2018) to constrain the training data extraction to three adjacent horizon slices: one at my horizon of interest, one above, and one below, amounting to a total of about three million data samples. Parameters of attribute computations and unsupervised training data extraction are listed in Appendix E. I then perform z-score normalization and logarithmic transformation on the training data. Mathematical descriptions of the z-score normalization and my logarithmic transformation are shown in Appendices A and B.

The original input data distributions are shown in Figure 4.5a. All six input attributes are non-negative and have skewed distributions. Note that z-score normalization does not change the shapes of the original distributions, but rather only shifts and stretches them (Figure 4.5b). In contrast, logarithmically transformed data have much more symmetrical shapes that closely resemble the “bell” curve of an ideal Gaussian distribution (Figure 4.5c).

I then feed the normalized training data to different unsupervised classification algorithms, including two projections techniques (PCA and ICA) and two clustering techniques (SOM and GTM), using the exact same parameter configuration for both z-score normalization and logarithmic transformation. Depending on whether the classification generates two or three output components, I perform 2D color crossplot or RGB blending (Figure 4.6) to display the results.

Principal Component Analysis (PCA)

The purpose of PCA is to find an orthogonal coordinate system that best captures data variation and to project the input attributes onto this new coordinate system (Figure 4.7). Mathematically, PCA is equivalent to first computing the covariance matrix of the input attributes, then finding the eigen vectors and eigen values of the covariance matrix, and finally performing matrix multiplication between the input data and the eigen vectors to obtain the principal components. Each eigen vector represents the direction of a principal coordinate axis, whereas each eigen value represents the amount of data variation along an axis. I choose three output principal components corresponding to the three largest eigen values for my analysis to capture the majority of the data variation.

Figure 4.8 shows the 2D histograms of three output principal components. Each 2D histogram corresponds to a pair of principal components. To reduce the effect of extreme values on the histogram displays, I clip 5% data at the extreme negative and 5% data at the extreme positive of each principal component. I observe that logarithmic transformation produces 2D histograms that are more symmetric, circular, and diffuse than those generated from z-score normalization.

The final RGB blended images of three output principal components are shown in Figure 4.9. Overall, the colors in the logarithmically transformed result are balanced, while z-score normalization is biased toward the green (2nd component). Zooming in the yellow box in the logarithmically transformed image, I can distinguish between the magenta crevasse splay (marked by a magenta arrow) and the orange crevasse splays (marked by orange arrows), which I hypothesize are composed of different types of sediment. In the z-score normalized image, those splays appear to a more uniform purple. A close-up inspection of the red box in the logarithmically transformed image reveals small, yellowish channels that make up the internal structure of a larger orange crevasse splay. In the z-score normalized image, those details are absent, and the splay appears as a homogeneous dark red patch.

Independent Component Analysis (ICA)

One assumption of PCA is that the data can be represented by a single multi-dimensional ellipsoidal "cloud". However, in practice, a data distribution can be composed of different clouds with different shapes, sizes, and orientations, making orthogonal principal components unable to fully capture the variation among all the data clouds (Figure 4.10). By finding a non-orthogonal coordinate system in an iterative manner, using higher order statistics, ICA aims to further separate different seismic facies and capture even more data variation (Lubo-Robles and Marfurt, 2019). To make it consistent with my PCA workflow, I specify three output independent components for my ICA model. Detailed parameter configuration of our ICA implementation is listed in Appendix E.

Similar to PCA, I construct a 2D histogram per each pair of output independent components (Figure 4.11). ICA 2D histograms exhibit the same phenomenon that I observed in PCA 2D histograms: logarithmic transformation produces 2D histograms that are more symmetric and diffuse than those generated from z-score normalization. Yet, there is a critical difference between ICA and PCA 2D histograms. In PCA, even though the shapes of the data "clouds" are different between logarithmic transformation and z-score normalization, there is still a common pattern between a 2D histogram produced by logarithmic transformation and a corresponding one produced by z-score normalization. For example, the data cloud in Figure

4.8c is somewhat similar to the data cloud in Figure 4.8f. However, in ICA, logarithmically transformed histograms show radically different data clouds than the corresponding z-score normalized histograms. For example, the data cloud in Figure 4.11a has a totally different shape, size, and orientation from the data cloud in Figure 4.11d. This is likely because the rank and polarity of the output independent components are undefined (Lubo-Robles and Marfurt, 2019). In other words, unlike PCA where principal components are ordered by their corresponding eigen values, it's impossible to tell before-hand that an independent component captures more data variation than another. Using a different normalization scheme somehow changes the polarity and possibly the order of the independent components, causing ICA algorithm to converge to completely different results.

The radical histogram difference between logarithmic transformation and z-score normalization is also reflected in the final RGB blended images of three output independent components (Figure 4.12). The images exhibit completely different color gamut. Level of details and color contrast in logarithmically transformed and z-score normalized ICA images are similar to each other and are both comparable to the logarithmic PCA image in Figure 4.9b. It is difficult to tell which normalization scheme is better, even in the zoomed-in sections. Therefore, for ICA, there is little difference in using a more sophisticated logarithmic transformation rather than the simpler z-score normalization.

Self-Organizing Maps (SOM)

Originally used in medical research for gene pattern recognition, SOM is now widely adopted as a clustering algorithm for seismic facies thanks to its relatively fast computation and its capability to show the proximity of one cluster to another (Zhao et al., 2015). SOM algorithm works by defining an initial grid of prototype vectors (neurons) on a plane defined by the first two eigenvectors called the manifold, which in turn are mapped to a 2D color table. As the process iterates, the manifold deforms to better represent the input training data (Figure 4.13). Clustering is done by finding the closest prototype vector to the data vector at each voxel. In my SOM implementation, I output not only the class number, but also the coordinates of each class on the two axes of the latent space, thereby allowing me to use commercial crossplotting tools to define features of interest. Parameter configuration of my SOM model is listed in Appendix E.

Figure 4.14 shows the crossplot images of the two SOM components along with the corresponding 2D histograms. The crossplot of the z-score normalized SOM components requires a strong data clipping in order to have a similar color contrast with the non-clipped crossplot of the logarithmically transformed SOM components. White pixels in the z-score normalized image represent clipped extreme data points. Appendix C explains what happens if I under-clip or over-clip the data for 2D crossplot. I observe that z-score normalized image, even after a strong clipping, is a lot greener than the logarithmically transformed image, due to the data distribution being skewed to the lower left corner of the z-score normalized 2D histogram. In contrast, the logarithmic transformation produces a much more color-balanced image and a more diffuse 2D histogram, allowing interpreters to better delineate different facies, such as distinguishing the yellow and orange crevasse plays from the green flood plain or mud-filled channels.

Generative Topographic Mapping (GTM)

Similar to SOM, GTM also aim to generate and iteratively deform a gridded manifold to best fit the input training data. However, the difference is in how the clustering is done. As noted by Zhao et al. (2015), while SOM "snaps" a data point to the closest grid node to generate a cluster, GTM put a Gaussian probability distribution function at each and every grid node on the manifold to estimate the contribution of all the grid nodes to a data point (Figure 4.15). Parameter configuration of our GTM model is listed in Appendix E.

My GTM implementation results in two GTM components, which I display using crossplots (Figure 4.16). Unlike SOM, no clipping is needed for both z-score normalized and logarithmically transformed GTM components. Again, I observe a similar phenomenon in the 2D histograms: the data are more evenly distributed in the logarithmic 2D histogram, while the z-score normalization exhibits very high data concentration within a single cell just below the center of the 2D histogram, causing the z-score crossplot image to have more green and greyish yellow pixels than the logarithmic crossplot image. Although the z-score crossplot image appears to enhance small details, these "details" are not geological, but rather random noise, thus causing the actual geological features (such as crevasses splays and channels) to appear more segmented than those in the logarithmic crossplot image.

Case Study 2: Supervised Classification of Salt in The Eugene Island Mini-basin, Gulf of Mexico

Geologic Setting

The Eugene Island mini-basin (Figure 4.17) contains one of the largest oil and gas fields in the northern part of the Gulf of Mexico, offshore Louisiana. The development of the Eugene Island mini-basin occurs in relatively recent geologic time, during the Pliocene-Pleistocene (Joshi and Appold, 2016), and consists of three phases: (1) pro-delta, (2) proximal deltaic, and (3) fluvial (Alexander and Flemings, 1995). During the pro-delta phase, rapid deposition of shales and turbidite caused the underlying Miocene salt to withdraw from the basin and buoy upward, leading to the creation of salt diapirs (Joshi and Appold, 2016). Salt continued to mobilize during the proximal deltaic phase, creating more accommodation space for low-stand shelf-margin deposition of alternating sand-shale sequences (Alexander and Flemings, 1995). At the final fluvial stage, salt withdrawal ended, leading to the formation of erosional unconformities during low-stands and the deposition of shallow-water deltaic and fluvial sand and shale during high-stands (Joshi and Appold, 2016). Most of the major reservoirs are laterally extensive sand sheets deposited during the proximal deltaic phase, which are sealed by laterally extensive shale layers on the top and by salt diapirs on the sides (Alexander and Flemings, 1995).

Methods

Figure 4.18 shows my supervised classification workflow. As with unsupervised classification, the first step is to select input attributes relevant to the supervised classification of salt. Lubo-Robles et al. (2021) performed an exhaustive search to find the best combination of input attributes and the best corresponding parameters for a PNN supervised classification. The authors found that a combination of seismic attributes given by coherence, gray-level cooccurrence matrix (GLCM) contrast, total energy, and dip deviation yield the lowest global validation error. However, as I implemented the exhaustive search to different normalization schemes, I found that each normalization scheme leads to a different optimal combination of input attributes (see Appendix D for more details). Since my research focuses on data normalization, I want my supervised classification workflow to be consistent with different normalization schemes. This requires me to use the same set of input attributes for all normalization schemes. I notice that the coherence attribute is found in all four normalization schemes' optimal input combinations. Therefore, to simplify my demonstration, I decide to use coherence attribute as the sole input for all of my supervised classifications.

In the next step, I follow the work by Qi et al. (2016) and apply a Kuwahara median filter to the input coherence attribute in order to smooth the internal noise of the salt diapirs, while sharpening the boundaries between the salt diapirs and the surrounding sedimentary layers (Figure 4.19). Parameters of the Kuwahara median filter are listed in Appendix E.

I pick different polygons inside and outside a salt diapir to denote salt and not-salt training data (Figure 4.20). After that, I extract the training samples within those polygons from the Kuwahara-filtered coherence attribute volume. Parameters of supervised training data extraction are listed in Appendix E.

Figure 4.21 illustrates the difference between bulk normalization scheme (in which I compute and apply the same normalization to the training data of all classes) and per-class normalization scheme (in which I normalize the training data of each class separately). Together with logarithmic transformation and z-score normalization, I perform a total of four different normalization schemes on the training data: (a) logarithmic bulk normalization, (b) logarithmic per-facies normalization, (c) z-score bulk normalization, and (d) z-score per-facies normalization. I then feed the four normalized training datasets into the PNN algorithm to generate four models, and finally classify the entire volume using the generated models.

Probabilistic Neural Network (PNN)

Essentially, PNN algorithm performs supervised classification by estimating the probability density functions of different classes or facies (hence the name **probabilistic** neural network), most commonly by approximating a Gaussian distribution to the input training data of each class (Specht, 1990; Masters, 1995; Lubo-Robles et al., 2021). I summarize the mathematical description of PNN algorithm in Appendix D. Again, to make sure my workflow is consistent, I set PNN smoothing parameter r=1.0 in all four normalization schemes.

Figure 4.22 shows the prediction results of PNN supervised classification using four normalization schemes. Green corresponds to salt facies, red corresponds to not-salt facies, and black corresponds to muted and no-permit zones. I observe several green patches (yellow ellipses) outside the salt diapir in the prediction result associated with the logarithmic bulk normalization scheme (Figure 4.22a), which do not appear in the prediction result associated with the z-score bulk normalization scheme (Figure 4.22c). The faulted region above the salt diapir (blue ellipse) is correctly classified as the not-salt facies in the prediction results associated with per-facies normalization (Figure 4.22b and Figure 4.22d), but are misclassified as salt facies in the prediction results associated with bulk normalization (Figure 4.22a and Figure 4.22c). Within per-facies normalization scheme, logarithmic transformation and z-score normalization appear to yield very similar prediction results (Figure 4.22b and Figure 4.22d). However, as I display the salt probability results (Figure 4.23) corresponding to the prediction results in Figure 4.22, using the same probability scale from 0.0 to 1.0, I notice that logarithmic per-facies normalization scheme (Figure 4.23b) yield a dimmer, lower-contrast image of salt probability than z-score per-facies normalization scheme (Figure 4.23d). This means the PNN algorithm is less confident in classifying salt facies with logarithmic per-facies normalization scheme than with z-score per-facies normalization scheme.

Discussion

Given such radical differences in machine learning results using different data normalization scheme, there are several questions to be addressed. Why did logarithmic transformation improve unsupervised classification but not supervised classification? What is the reason behind the enhancement made by per-class normalization scheme to supervised classification? The answers to these questions are given by a fundamental difference between unsupervised and supervised classifications (other than the obvious need for human interaction).

Unsupervised Essence: Color Mapping

In unsupervised classification, the manner in which colors are mapped to the results is critical. In fact, the whole idea behind unsupervised classification is to “paint” each and every single data point in such a way that points having similar values (i.e. belonging to the same cluster) have similar colors. However, if a group of data points is assigned with one single color, then the differences among that group of points are lost when displaying the classification results. The loss of color differences is illustrated in the SOM and GTM crossplots using z-score normalization scheme (Figure 4.14a and Figure 4.16a). The 2D histograms of z-score SOM and GTM show that the data are highly concentrated within a cell, thus “painting” a large chunk of data points with the same color in the crossplots. In contrast, 2D histograms of logarithmic SOM and GTM (Figure 4.14b and Figure 4.16b) show a more diffuse distribution, highlighting subtle variations of data points near the center of the data distribution, thus allowing more colors and fine detail to appear in the crossplots.

Another illustration of color mapping in unsupervised classification is the RGB blending process of PCA results. Examine the first principal component projection plotting against the red color channel (Figure 4.24). The histograms are clipped in the same way as in RGB blending of the PCA projections: only 90% of the data distribution is shown, while the 5% data at the extreme left and the 5% data at the extreme right of the distribution are clipped. Even at such a strong clipping, z-score data distribution is skewed to the right and has a narrower shape compared to logarithmic data distribution, meaning a large portion of the data is concentrated near the peak of the distribution. Thus, fewer colors are used to map the area about the peak of the distribution. In contrast, the logarithmic data distribution exhibits a much wider and symmetric shape than the z-score normalization. Thus, more colors are mapped near the peak of the distribution. Essentially, the logarithmic transformation “squeezed” the extreme positive data points to the center and “stretched” the left side of the data distribution, thereby increasing the resolution near the peak of the distribution while reducing the resolution of the extrema. Fortunately for me, the subtle geological detail of the turbidite channel system resides near the peak of the distribution and thus was better resolved using the logarithmic transformation. I anticipate that logarithmic transformation would have produced worse unsupervised classification results than z-score normalization if my analysis had instead focused on resolving the extreme data points.

Supervised Essence: Shapes of Clusters and Distances among Clusters

Unlike unsupervised classification’s dependence on color mapping, supervised classification relies exclusively on the ability of the internal algorithm to distinguish among different clusters. Whether the clusters are distinguishable or not, in turn, heavily depends on the shapes of the clusters and the distances from one cluster to another. To illustrate the differences in the shapes of the clusters and the distances among the clusters for different normalization schemes, I plot the histograms of PNN's input coherence attribute in four cases (corresponding to the four PNN salt probability images shown in Figure 4.23): (a) logarithmic bulk transformation, (b) logarithmic transformation based on training data of salt facies, (c) z-score bulk normalization, and (d) z-score normalization based on training data of salt facies (Figure 4.25). These histograms display 100% of the data distribution and are resized to the same horizontal scale. Because salt is incoherent, the salt cluster have lower coherence values than the not-salt cluster, and thus the salt cluster falls to the left, while the not-salt cluster falls to the right of the histograms.

I observe that with the bulk normalization scheme, the distance between salt and not-salt clusters are approximately the same in z-score and logarithmic histograms, but the shape of the salt cluster associated with logarithmic transformation (Figure 4.25a) is significantly wider than that associated with z-score normalization (Figure 4.25c). This is likely due to an inherent characteristic of logarithmic function, which tends to stretch the data points whose values are close to zero (which is where the salt cluster is located). With the logarithmic transformation, the wider shape of the salt cluster makes it harder for the supervised algorithm to discriminate salt from not-salt facies, thus causing many coherent data points to have high salt probability (yellow ellipses in Figure 4.22a).

In the per-class normalization scheme, the distance between salt and not-salt clusters is shorter in the logarithmic histogram (Figure 4.25b) than in the z-score histogram (Figure 4.25d), because logarithmic transformation tends to "squeeze" the right side of the data distribution, thereby moving the not-salt cluster closer to the salt cluster. The shorter distance between the two clusters means the supervised classification is less confident in distinguishing the two facies, causing an overall dimmer salt probability image associated with logarithmic transformation (Figure 4.23b).

On the other hand, for both logarithmic and z-score normalization, the distance between the salt and not-salt clusters associated with per-class normalization scheme (Figure 4.25b and Figure 4.25d) is much greater than that associated with bulk normalization scheme (Figure 4.25a and Figure 4.25c). This is because bulk training data in supervised classification is a collection of different clusters and thus, by definition, has a greater variation than the individual clusters. By normalizing the training data per each seismic facies, the distance between the two clusters become much larger, allowing the supervised algorithm to distinguish between different classes with a much higher confidence, hence significantly higher contrast in the PNN probability result (Figure 4.23b and Figure 4.23d).

Conclusions

My verdict is that data normalization, despite being a hidden step that is most often set to default in a machine learning workflow, deserves more attention. Different data normalization schemes can generate significantly different classification results. Logarithmic transformation produces unsupervised classification results with more subtle details and higher color contrast than z-score normalization because the logarithmic function tends to collapse the extreme positive data points while simultaneously expanding the near-zero values, effectively allowing more colors to be mapped around the peak of a distribution – where the data concentrates the most. However, for the same reason, the logarithmic transformation tends to shift the clusters of different classes closer together, thereby reducing the confidence of a supervised algorithm to distinguish between different classes, compared to z-score normalization. Per-class normalization scheme yields significantly greater distances among different clusters than the typical bulk normalization scheme, thus producing much higher contrast in the probability results. Therefore, it is critically important to try different normalization schemes on the input training data, carefully analyzing the distributions of the training data and the classification results, and comparing side-by-side the final displays in order to obtain the most optimal normalization scheme for a specific machine learning workflow.

Acknowledgments

I highly appreciate the New Zealand Petroleum and Minerals (NZPAM) for providing the Canterbury Basin dataset, as well as the U.S. National Archive of Marine Seismic Surveys (NAMSS) and the U.S. Geological Survey (USGS) for providing the Eugene Island dataset. Financial support was through the industry sponsors of the Attribute-Assisted Seismic Processing and Interpretation (AASPI) consortium. Finally, I would like to express my gratitude to Mr. Ichie Yasutaka at JX Nippon Oil and Gas Exploration Corporation in Japan for his advices and suggestions.

Appendix A: Derivation of Logarithmic Transformation

Expectation vs. Reality

Mathematically, a logarithmic transformation refers to the application of the logarithmic function, $\log_k(x)$, to the input data. My goal is to use a logarithmic transformation to reshape a non-negative, right-skewed data distribution to be more symmetric and thus closer to an ideal "bell"-shaped Gaussian distribution (Figure 4.A-1). However, if I directly apply $\log_k(x)$ to the input data, I would instead end up with an even more asymmetric data distribution, only this time it's left-skewed, with a very long left "tail" stretching to negative infinity. This undesired result occurs because $\log_k(x)$ approaches negative infinity as x goes to zero (Figure 4.A-2). Furthermore, $\log_k(x)$ is undefined for negative values of x, the results of which in a computer are commonly represented by NaN, (abbreviation for Not-a-Number), which is detrimental to all subsequent computations. To find a better way to apply the logarithmic transformation, I need to go back to the basics, where I will first generalize linear transformations, including the well-known z-score normalization.

Z-score Normalization Revisited: Generalizing a Linear Transformation

Z-score, or standard score, is by far the most widely used data normalization scheme. The process involves subtracting the mean from the data, then divide the result by the data's standard deviation:

$$y_n = \frac{x_n - \mu}{\sigma}, \tag{A-1}$$

where

x is the input data vector of length N: $\mathbf{x} = [x_1, x_2, \ldots, x_N]$,

μ is the mean (or the arithmetic average) of **x**: $\mu = \frac{\sum_{n=1}^{N} x_n}{N}$, (A-2)

σ is the standard deviation of **x**: $\sigma = \sqrt{\frac{\sum_{n=1}^{N}(x_n - \mu)^2}{N}}$, and (A-3)

y is the normalized data vector: **y** = [y_1, y_2, …, y_N].

Equation A-1 can be rewritten into a different form, in which the subtraction is equivalent to an addition of negative mean, and the division is equivalent to a multiplication of the reciprocal of the standard deviation:

$$y_n = \frac{1}{\sigma}[x_n + (-\mu)] \quad \text{(A-4)}$$

Equation A-4 can be further generalized as:

$$y_n = b(x_n + a) \quad \text{(A-5)}$$

Where $a = -\mu$ is the shift factor, and $b = \frac{1}{\sigma}$ is the scale factor of z-score normalization. There are only two parameters (shifting and scaling) associated with a linear transformation, and any normalization scheme that has the form of Equation A-5 is a linear transformation.

The goal of z-score normalization is to transform the input data into one with a mean of zero and a standard deviation of one, similar to a *normal* distribution (hence the term “normalization”). Z-score normalization assumes the input data distribution has approximately the same “bell” shape of a normal distribution. However, in practice, the majority of seismic attributes are non-negative, exhibiting skewed and asymmetrical distribution that are poorly represented by “bell” shape assumption. The linear z-score normalization neither changes the skewness nor the asymmetricity of a distribution to more closely resemble a normal distribution.

Generalizing the Logarithmic Transformation

Based on the generalized formula of linear transformation in Equation A-5, a fully generalized logarithmic transformation is equivalent to first linearly transforming the input, followed by the application of the logarithmic function and then followed by yet another linear transformation in the logarithmic domain:

$$y_n = c\{\log_k[b(x_n + a)] + d\} \tag{A-6}$$

where:

$\log_k()$ is the logarithmic function of base k,

d is the shift factor in the logarithmic domain, and

c is the scale factor in the logarithmic domain.

Recalling from high-school algebra, there are three important properties of the logarithmic functions. First, the logarithmic function of base k can be rewritten using the natural logarithmic function:

$$\log_k(u) = \frac{\ln(u)}{\ln(k)}. \tag{A-7}$$

Second, the power rule of the logarithm states that

$$\ln(u^m) = m\ln(u). \tag{A-8}$$

Third, and most importantly, the logarithm of a product is the sum of the individual logarithms:

$$\ln(uv) = \ln(u) + \ln(v). \tag{A-9}$$

Using equations A-7, A-8 and A-9, I rewrite equation (A-6) as:

$$y_n = c\left\{\frac{\ln[b*(x_n+a)]}{\ln(k)} + \frac{\ln(k^d)}{\ln(k)}\right\} = \frac{c}{\ln(k)}\{\ln[b(x_n + a)] + \ln(k^d)\}, \text{ thus}$$

$$y_n = \frac{c}{\ln(k)}\{\ln[k^d b(x_n + a)]\} \tag{A-10}$$

Note that in equation A-10, $\frac{c}{\ln(k)}$ can be viewed as a single factor of scaling in the logarithmic domain, while $k^d b$ can be viewed as a single factor of scaling in the original data domain. This simplification occurs because shifting in the logarithmic domain can expressed as scaling in the original data domain, as implied by equation A-9. Thus, the generalized formula for logarithmic transformation can be rewritten as follow:

$$y_n = c\{\ln[b(x_n + a)]\} \tag{A-11}$$

Essentially, I have reduced the number of parameters in the logarithmic transformation from five (*a*, *b*, *c*, *d*, and *k* in equation A-6) to three (*a*, *b*, and *c* in equation A-11). Even at three parameters, there are still unlimited possibilities for a logarithmic transformation. I need to define some constrains for logarithmic transformations in order to reshape the input data distribution as close to a Gaussian distribution as possible.

Finding Optimal Logarithmic Transformation Parameters

One way to reframe the Gaussian reshaping problem is to simplify the shape of a data distribution into three anchor points: the left (L), the right (R), and the peak (P) (Figure 4.A-3a). In my implementation, in order to avoid the effects of the extreme values, I retain only 95% of the data distribution by defining x_L as the 2.5% percentile instead of the minimum value, and x_R as the 97.5% percentile instead of the maximum value. The goal of the Gaussian reshaping process is that after the logarithmic transformation, the transformed left and right anchor points are symmetric about zero, while the peak of the transformed distribution is located exactly at zero:

$$\begin{cases} y_L = -y_R \\ y_P = 0 \end{cases}. \tag{A-12}$$

Substituting equation A-11 into equations A-12, we have:

$$\begin{cases} c\{\ln[b(x_L + a)]\} = -c\{\ln[b(x_R + a)]\} \\ c\{\ln[b(x_P + a)]\} = 0 \end{cases} \rightarrow \begin{cases} a = \frac{x_P^2 - x_L x_R}{x_L + x_R - 2x_P} \\ b = \frac{1}{x_P + a} \end{cases} \tag{A-13}$$

Using the derived parameters in equation bracket A-13, I expect the transformed data distribution to be a nice symmetrical one (Figure 4.A-3b). Instead, I end up with a slightly skewed distribution (Figure 4.A-3c). As logarithmic transformation reshapes the data distribution, it also moves the relative position of the peak! In other words, the peak of the transformed distribution is not at the same position with the peak of the original distribution. Therefore, I need to compute the transformation parameters iteratively by recomputing the peak anchor point of the original distribution x_P from the peak of the transformed distribution y_P at every iteration j:

$$x_P^{(j)} = \frac{\exp\left(y_P^{(j)}\right)}{b^{(j)}} - a^{(j)}. \tag{A-14}$$

As I soon find out, reality is far from expectation. Using equation A-14 in an iterative manner, the peak of the transformed distribution does not converge in some cases, but instead "jumps" left and right alternatively, and may diverge in some extreme cases. My final remedy to this peak-jumping issue is to compute the average peak of all iterations up to the current *j*th iteration:

$$x_P^{(j)} = \frac{1}{j+1}\left\{x_P^{(0)} + \sum_{m=1}^{j}\left[\frac{\exp\left(y_P^{(m)}\right)}{b^{(m)}} - a^{(m)}\right]\right\}. \tag{A-15}$$

Where $x_P^{(0)}$ is the actual peak of the original data distribution. Although equation A-15 does not provide a mathematically accurate convergence of the peak anchor point, the computation is relatively fast, is guaranteed to converge, and results in a good approximation to a Gaussian distribution. In all of my tests, with only M=100 iterations, the peak was able to converge within at least four to five significant digits. With about 3 million training samples in my project, it took only 5 seconds to finish 100 iterations, without parallelization, using Fortran code.

Even though the peak of the reshaped distribution is centered about zero, the mean of the transformed distribution is not necessarily at the same position as the peak. Therefore, at the end of the iterative computation, I need to add another term to the linear scale factor *b* in equation bracket A-13 in order to shift the mean of the reshaped distribution to zero:

$$\tilde{b} = b^{(M)}\exp\left(-\mu_{LOG}^{(M)}\right) = \frac{\exp\left(-\mu_{LOG}^{(M)}\right)}{x_P^{(M)}+a^{(M)}} \tag{A-16}$$

where:

M is the last iteration, and

$\mu_{LOG}{}^{(M)}$ is the mean of the last iteration's result: $\mu_{LOG}^{(M)} = \frac{\sum_{n=1}^{N}\ln\left[b^{(M)}\left(x_n+a^{(M)}\right)\right]}{N}$ (A-17)

Note that in equation bracket A-13, there is no solution for the logarithmic scale factor *c*. Instead, *c* is defined as the reciprocal of the standard deviation of the reshaped data, in order to ensure the final transformed data has a standard deviation of one. Additionally, if the distribution is flipped (i.e. the linear scale factor $\tilde{b}$ is negative), then the sign of the logarithmic scale factor *c* needs to be reversed as well:

$$c = \begin{cases} \frac{1}{\sigma_{LOG}^{(M)}}\ if\ \tilde{b} > 0 \\ undefined\ if\ \tilde{b} = 0 \\ -\frac{1}{\sigma_{LOG}^{(M)}}\ if\ \tilde{b} < 0 \end{cases} \tag{A-18}$$

where $\sigma_{LOG}{}^{(M)}$ is the standard deviation of the last iteration's result:

$$\sigma_{LOG}^{(M)} = \sqrt{\frac{\sum_{n=1}^{N}\left\{\ln\left[b^{(M)}\left(x_n+a^{(M)}\right)\right]-\mu_{LOG}^{(M)}\right\}^2}{N}} \tag{A-19}$$

Using equations A-11, A-13, A-16, and A-18, I summarize the logarithmic transformation in the following pseudocode:

```
Function log_transform(x)
•  sort(x)
•  $x_P$=peak(x)
•  $x_L$=percentile_2.5(x)
•  $x_R$=percentile_97.5(x)
• sum=$x_P$
• Loop j=1 to 100:
  o a=($x_P$^2-$x_L$*$x_R$)/($x_L$+$x_R$-2*$x_P$)
  o b=1.0/($x_P$+a)
  o If b==0 or b==±∞ or a==±∞:
   ➢ Fall back to z-score:
   ➢ a=-mean(x)
   ➢ b=1.0/standard_deviation(x)
   ➢ y=b*(x+a)
   ➢ Return y, a, b
  o End If
  o y=ln(b*(x+a))
  o $y_P$=peak(y)
  o sum=sum+exp($y_P$)/b-a
  o $x_P$=sum/(j+1)
• End Loop j
• b=b*exp(-mean(y))
• c=1.0/standard_deviation(y)
• If (b<0): c=-c
• y=c*ln(b*(x+a))
• Return y, a, b, c
End Function log_transform
```

Appendix B: Finding The Peak (or The Mode) of a Data Distribution

As described in Appendix A, one of the critical tasks in the computation of logarithmic transformation parameters is to find the peak of a data distribution. The peak (in statistics, more precisely denoted as the mode) of a data distribution is the location of greatest data density. Usually, the process of determining the peak of a data distribution involves segregating the distribution into multiple histogram columns and then locating the highest-valued column (Figure 4.B-1). The precision of the peak depends on the width of each histogram column. Furthermore, if there is a spike in a data distribution (where multiple data points having exactly the same value, such as zero samples belonging to a dead trace or a muted zone), this method will most likely pick the spike instead of the desired peak.

In this appendix, I present an alternative method to find the peak of a data distribution based on the binary search algorithm described by Lin et al. (2019). The process starts by limiting the search between the 15% percentile and the 85% percentile of a distribution with the assumption that, in most cases, the peak falls within such a range around the distribution center (Figure 4.B-2). I then divide this range into two halves that have exactly the same width. Next, I choose the denser half (i.e. the half representing the greater number of data points) and start dividing it into another two halves, and repeat. The assumption is that the peak of a distribution will always reside in the denser half, and thus by successively choosing and dividing the denser half, eventually I will come down to two halves with exactly one data point in each. At the end, I simply take the average of these two points to represent the peak of the distribution.

As with every other algorithm, there are pros and cons to this binary-search method to find the peak of a data distribution. One of the algorithm's positive features is that it is highly precise, on the order of one half the distance between two adjacent data points. It is inherently fast because binary search algorithms have a worst-case-scenario efficiency of O(log *N*) (Lin et al., 2019). And since it's limiting the search to the center part of the distribution, it is robust against zero-value spikes caused by dead traces and muted zones, which are usually located at the extreme left of the distributions of non-negative seismic attributes, provided that the spikes do not constitute a significant portion of the data (i.e. less than 15%).

In terms of negative features, the algorithm requires the data to be sorted in ascending order (though in logarithmic transformation, sorting is already required to estimate the 2.5% percentile and 97.5% percentile for the left and right anchor points). A more serious issue arises when, after dividing a denser half into two, the resulting two halves have exactly the same data count of a large quantity (Figure 4.B-3). In this case, the midpoint between the two halves is set as the peak of the distribution and the binary search would stop, even though the actual peak might still be located within either of the two halves. This is rare, but not impossible, and becomes more likely when the total number of data points is relatively small (on the order of thousands or less) and the data distribution contain more than one cluster of similar sizes. To fix this issue, I need to further divide the two halves into four quarters and choose the quarter with the maximum data count, assuming that the four quarters have different data counts and the actual peak of the distribution is in the biggest quarter.

I summarize my peak binary search method in the following pseudocode:

```
Function peak_binary_search(x)
• n=length(x)
• If not_sorted(x): sort(x)
• j_mid=index_percentile_15(x)
• j_right=index_percentile_85(x)
• left_count=0
• right_count=j_right-j_mid+1
• Loop While right_count>1:
  o Loop While left_count≠right_count:
    ➢ If left_count<right_count:
      ✓ j_left=j_mid
    ➢ Else
      ✓ j_right=j_mid-1
    ➢ End If
    ➢ mid_val=0.5*(x(j_left)+x(j_right))
    ➢ Loop j_mid=j_left+1 to j_right:
      ✓ If x(j_mid)>=mid_val:
      ✓ Exit Loop j_mid
    ➢ End Loop j_mid
    ➢ left_count=j_mid-j_left
    ➢ right_count=j_right-j_mid+1
  o End Loop While left_count≠right_count
  o If right_count>1:
    ➢ mid_left_val=0.5*(x(j_left)+x(j_mid-1))
    ➢ mid_right_val=0.5*(x(j_mid)+x(j_right))
    ➢ Loop j_mid_left=j_left+1 to j_mid-1:
      ✓ If x(j_mid_left)>=mid_left_val:
      ✓ Exit Loop j_mid_left
    ➢ End Loop j_mid_left
    ➢ Loop j_mid_right=j_mid+1 to j_right:
      ✓ If x(j_mid_right)>=mid_right_val:
      ✓ Exit Loop j_mid_right
    ➢ End Loop j_mid_right
    ➢ left_1_count=j_mid_left-j_left
    ➢ left_2_count=j_mid-j_mid_left
    ➢ right_1_count=j_mid_right-j_mid
    ➢ right_2_count=j_right-j_mid_right+1
    ➢ max_left=max(left_1_count,left_2_count)
    ➢ max_right=max(right_1_count,right_2_count)
    ➢ If max_left<max_right:
      ✓ j_left=j_mid
      ✓ j_mid=j_mid_right
      ✓ mid_val=mid_right_val
      ✓ left_count=right_1_count
      ✓ right_count=right_2_count
    ➢ Else:
      ✓ j_right=j_mid-1
      ✓ j_mid=j_mid_left
      ✓ mid_val=mid_left_val
      ✓ left_count=left_1_count
      ✓ right_count=left_2_count
    ➢ End If max_left<max_right
  o End If right_count>1
• End Loop While right_count>1
• Return mid_val
End Function peak_binary_search
```

Appendix C: Data Clipping in Crossplot

This appendix shows the effect of different data clipping strategies on the 2D crossplot between two components where I will use the crossplot of two z-score SOM output components as an example. Figure 4.C-1 shows the SOM crossplot image without any data clipping. The entire range from the minimum value to the maximum value of each component is used in the construction of the crossplot. I observe that the image is very pale (i.e. has very low color contrast) and is mostly cyan. I can barely see any geological detail. This lack of contrast is because the data are mapped mostly to the upper left corner of the 2D histogram, which corresponds to a variety of cyan colors.

More careful inspection of the 2D histogram of full-range SOM crossplot in Figure 4.C-1 reveals that the majority of the data distribution falls within -2.6 to 3.0 for SOM component 1 and -3.0 to 3.0 for SOM component 2. By clipping the data using these ranges, I end up with Figure 4.C-2. The color contrast is good, but there are too many white pixels representing clipped data values (or “outliers”).

Therefore, I slightly increase the ranges of SOM components for the crossplot, from -2.6 to 4.0 for SOM component 1 and -3.5 to 3.5 for SOM component 2 (as indicated in Appendix E), in order to reduce the number of white pixels while maintaining most of color contrast. The result is Figure 4.C-3, which is exactly the same with Figure 4.14a. This strategy does require some trial-and-error attempts to configure data clipping in order to obtain the (subjectively) best display of the 2D crossplot.

Appendix D: Mathematical Description of PNN Exhaustive Search

In this appendix, I summarize the mathematical equations of the probabilistic neural network (PNN) algorithm and the exhaustive search to determine the best combination of input attributes and the corresponding PNN parameters. Following Specht (1990), Masters (1995) and Lubo-Robles et al. (2021), the PNN algorithm estimates the probability density function of each class in the input supervised training data, most commonly by using the Gaussian function. The estimated density function of the *M*-element attribute data vector $\mathbf{x}_j$ at voxel *j*, with respect to class *k* is:

$$g_k(\boldsymbol{x}_j) = \frac{1}{N_k}\sum_{n=1}^{N_k} \exp\left[\sum_{m=1}^{M} \frac{\left(x_{jm}-u_{nmk}\right)^2}{r^2}\right] \qquad \text{(D-1)}$$

where:

N_k is the number of training samples assigned to class *k*,

M is the number of input attributes,

x_{jm} is the value of the *m*th attribute of the normalized input data vector at voxel *j*,

u_{jmk} is the value of the *m*th attribute of the *n*th normalized training sample belonging to class *k*, and

r is the smoothing parameter, which is also the only PNN parameter that requires further optimization.

$g_k(\mathbf{x}_j)$ is then normalized by the sum of estimated density functions for the input data vector $\boldsymbol{x}_j$ with respect to all *K* classes, resulting in the probability of sample $\mathbf{x}_j$ belonging to class *k* as:

$$P_k(\mathbf{x}_j) = \frac{g_k(x_j)}{\sum_{q=1}^{K} g_q(x_j)}. \qquad \text{(D-2)}$$

The probability of an input sample $\mathbf{x}_j$ belonging to class *k* depends on the proximity of $\mathbf{x}_j$ to all of the training samples $\mathbf{u}_{nk}$ belonging to class *k*. The class exhibiting the highest probability is defined as the predicted class of the input sample $\mathbf{x}_j$. A PNN model is just a collection of the smoothing parameters *r*, the normalized training samples $\mathbf{u}_{nk}$, and the normalization parameters. Note that for a per-class normalization scheme, I calculate the normalization parameters for class *k* using only the training samples belonging to class k. Then, I normalize both the input to-be-classified samples **x** and the training samples $\mathbf{u}_{nk}$ using class *k*'s unique normalization parameters before the probability computation.

To perform an exhaustive search using PNN algorithm, Lubo-Robles et al. (2021) iterated through all possible combinations of input attributes. For each combination of input attributes, the optimal smoothing parameter *r* is the one that yields the lowest global validation error:

$$E = \frac{1}{N_v} \sum_{j=1}^{N_v} e_k(x_j) \tag{D-3}$$

where:

N_v is the number of validation samples (i.e. blind-test samples), and $e_k(x)$ is a continuous error function (Masters, 1995) defined as:

$$e_k(x) = [1 - P_k(x)\,]^2 + \sum_{h \neq k} [P_h(x)]^2 \tag{D-4}$$

Lubo-Robles et al. (2021) found that a combination of coherence, GLCM contrast, total energy, and dip deviation attributes, with smoothing parameter r=0.1, yields the lowest global validation error of 0.01689. I follow Lubo-Robles et al. (2021) and apply their PNN exhaustive search for each of my normalization schemes and find that for different normalization schemes, there are different validation errors and hence a different best combination of input attributes with different smoothing parameters (Table 4.D-1).

Appendix E: Lists of Computing Parameters

For case study 1, Tables 4.E-1 to 4.E-6 list parameters of attribute computations, including dip attributes (which are required as input to define the orientation of the structure for almost of the other attributes), dip filtering, coherence, structural curvature, GLCM attributes, and spectral decomposition. Table 4.E-7 lists parameters of training data extraction. Tables 4.E-8 to 4.E-11 list parameters of individual algorithm's modeling and plotting (including PCA, ICA, SOM, and GTM).

For case study 2, Tables 4.E-12 to 4.E-14 list parameters of attribute computation, including dip, dip filtering, and coherence attributes. Tables 4.E-15 and 4.E-16 list parameters of Kuwahara median filter. Table 4.E-17 lists parameters of training data extraction and PNN modeling.

References

Alexander, F., and P. Flemings, 1995, Geologic Evolution of a Pliocene–Pleistocene Salt-Withdrawal Minibasin: Eugene Island Block 330, Offshore Louisiana: AAPG Bulletin, 79, no. 12, 1737–1756. doi: 10.1306/7834DEEA-1721-11D7-8645000102C1865D

Castro de Matos, M., K. J. Marfurt, and P. R. Schroeder Johann, 2010, Seismic interpretation of self-organizing maps using 2D color displays: Brazilian Journal of Geophysics (RBGf), **28**, 631-642. doi: 10.1590/S0102-261X2010000400008

Chopra, S. and K. J. Marfurt, 2014, Churning seismic attributes with principal component analysis: SEG Expanded Abstract, 2672-2676. doi: 10.1190/segam2014-0235.1

Chopra, S. and K. J. Marfurt, 2014, Seismic facies analysis using generative topographic mapping: SEG Expanded Abstract, 1390-1394. doi: 10.1190/segam2014-0233.1

Cozens, N., 2011, A study of unconventional gas accumulation in Dannevirke Series (Paleogene) rocks, Canterbury Basin, New Zealand: Master's thesis, Victoria University of Wellington.

Guo, H., S. Lewis, and K. J. Marfurt, 2008, Mapping multiple attributes to three- and four-component color models – a tutorial: Geophysics, **73**, W7-W19. doi: 10.1190/1.2903819

Hardisty, R. and B. Wallet, 2017, Unsupervised seismic facies from mixture models to highlight channel features: SEG Expanded Abstract, 2289-2293. doi: 10.1190/segam2017-17794438.1

Honorio, B. C. Z., A Crus Sanchetta, E. Pereira Leite, and A. Campane Vidal, 2014, Independent component spectral analysis: Interpretation, **2**, SA21-SA29. doi: 10.1190/INT-2013-0074.1

Joshi, A., and M. Appold, 2016, Potential of porosity waves for methane transport in the Eugene Island field of the Gulf of Mexico basin: Marine and Petroleum Geology, 75, 1-13. doi: 10.1016/j.marpetgeo.2016.04.005

Kim, Y., R. Hardisty, and K. J. Marfurt, 2019, Attribute selection in seismic facies classification: Application to a Gulf of Mexico 3D seismic survey and the Barnett Shale: Interpretation, **7**, SE281-SE297. doi: 10.1190/INT-2018-0246.1

Lin, A., et al., 2019, Binary search algorithm: WikiJournal of Science, **2**. doi:10.15347/WJS/2019.005

Lubo-Robles, D. and K. J. Marfurt, 2019, Independent component analysis for reservoir geomorphology and unsupervised seismic facies classification in the Taranaki Basin, New Zealand: Interpretation, **7**, SE19-SE42. doi: 10.1190/INT-2018-0109.1

Lubo-Robles, D., T. Ha, S. Lakshmivarahan, K. J. Marfurt, and M. Pranter, 2021, Exhaustive Probabilistic Neural Network for attribute selection and supervised seismic facies classification: Interpretation, 9, doi: 10.1190/INT-2020-0102.1.

Masters, T., 1995, Advanced algorithms for neural networks: John Wiley & Sons.

Meldahl, P., R. Heggland, B. Bril, and P. de Groot, 1999, The chimney cube, an example of semi-automated detection of seismic objects by directive attributes and neural networks: Part I; methodology: SEG Expanded Abstract, 931-934. doi: 10.1190/1.1821262

Poupon, M., K. Azbel, and G. Palmer, 1999, A new methodology based on seismic facies analysis and litho-seismic modeling The Elkhorn Slough field pilot project, Solano County, California: SEG Expanded Abstract, 927-930. doi: 10.1190/1.1821260

Qi, J., T. Lin, T. Zhao, F. Li, and K. J. Marfurt, 2016, Semisupervised multiattribute seismic facies analysis: Interpretation, **4**, SB91-SB106. doi: 10.1190/INT-2015-0098.1

Qi, J. and K. J. Marfurt, 2019, Nonparallelism attributes and data adaptive Kuwahara image processing: SEG Expanded Abstract, 1858-1862. doi: 10.1190/segam2019-3216022.1

Qi, J., B. Zhang, B. Lyu, and K. J. Marfurt, 2020, Seismic attribute selection for machine-learning-based facies analysis: Geophysics, **85**, O17-O35. doi: 10.1190/GEO2019-0223.1

Qi, X., J. Snyder, T. Zhao, K. J. Marfurt, and M. J. Pranter, 2019, Correlation of seismic attributes and geomechanical properties to the rate of penetration in the Mississippian Limestone, Oklahoma, in G. M. Grammer, J. M. Gregg, J. O. Puckette, P. Jaiswal, S. J. Mazzullo, M. J. Pranter, and R. H. Goldstein, eds., Mississippian Reservoirs of the Midcontinent: AAPG Memoir 122. doi: 10.1306/13632162M1163795

Roy, A., A. S. Romero-Pelaez, T. J. Kwiatkowski, and K. J. Marfurt, 2014, Generative topographic mapping for seismic facies estimation of a carbonate wash, Veracruz Basin, southern Mexico: Interpretation, **2**, SA31-SA47. doi: 10.1190/INT-2013-0077.1

Roden, R., T. Smith, T. and D. Sacrey, 2015, Geologic pattern recognition from seismic attributes: Principal component analysis and self-organizing maps: Interpretation, **3**, SAE59-SAE83. doi: 10.1190/INT-2015-0037.1

Roden, R., and C. W. Chen, 2017, Interpretation of DHI characteristics with machine learning: First Break, **35**, 55-63. doi: 10.3997/1365-2397.35.5.88069

Specht, D. F., 1990, Probabilistic neural networks: Neural Networks, 3, 109-118, doi: 10.1016/0893-6080(90)90049-Q

Strecker, U., and R. Uden, 2002, Data mining of 3D poststack attribute volumes using Kohonen self-organizing maps: The Leading Edge, **21**, 1032-1037. doi: 10.1190/1.1518442

Strecker, U., N. Derzhi, M. Carr, S. Knapp, M. Smith, R. Uden, G. Taylor, T. Taner, 2005, Why interpret with seismic attributes? Caveats, keynotes & a case study featuring multiple seismic attribute analysis in hydrothermal dolomite: SIPES 3-D Symposium, Sept. 13, 2005.

Sutherland, R., and G. Browne, 2003, Canterbury basin offers potential on South Island, New Zealand: Oil & Gas Journal, 101, 45–49.

Torres, E., R. Slatt, K. J. Marfurt, L. Infante, and L. Castillo, 2018, Identification of Potential Lacustrine Stratigraphic Intervals in the Woodford Shale, Oklahoma, Using Multi-Attribute 3-D Seismic Displays and a Supervised Neural Network: Geophysical Society of Houston Journal, **8**, 13-25. doi: 10.1190/segam2017-17783467.1

Verma, S., A. Roy, R. Perez, and K. J. Marfurt, 2012, Mapping high frackability and high TOC zones in the Barnett Shale: Supervised Probabilistic Neural Network vs. unsupervised multi-attribute Kohonen SOM: SEG Expanded Abstract. doi: 10.1190/segam2012-1494.1

Wallet, B. and T. Ha, 2019, Deep Learning Method for Latent Space Analysis: SPE Middle East Oil and Gas Show and Conference. doi: 10.2118/194889-MS

West, B., S. May, J. E. Eastwood, and C. Rosen, 2002, Interactive seismic facies classification using textural and neural networks: The Leading Edge, **21**, 1042-1049. doi: 10.1190/1.1518444

Zhao, T., V. Jayaram, and K. J. Marfurt, 2014, Lithofacies classification in Barnett Shale using proximal support vector machines: SEG Expanded Abstract. doi: 10.1190/segam2014-1210.1

Zhao, T., V. Jayaram, A. Roy, and K. J. Marfurt, 2015, A comparison of classification techniques for seismic facies recognition: Interpretation, **3**, SAE29–SAE58. doi: 10.1190/INT-2015-0044.1.

Zhao, T., J. Zhang, F. Li, and K. J. Marfurt, 2016, Characterizing a turbidite system in Canterbury Basin, New Zealand, using seismic attributes and distance-preserving self-organizing maps: Interpretation, **4**, SB79-SB89. doi: 10.1190/INT-2015-0094.1.

Zhao, T., F. Li, and K. J. Marfurt, 2017, Constraining self-organizing map facies analysis with stratigraphy: An approach to increase the credibility in automatic seismic facies classification: Interpretation, **5**, T163-T171. doi: 10.1190/INT-2016-0132.1

Zhao, T. and K. J. Marfurt, 2017, Different training sample selection strategies in unsupervised seismic facies analysis: SEG Expanded Abstract, 2132-2136. doi: 10.1190/segam2017-17739947.1

Zhao, T., F. Li, and K. J. Marfurt, 2018, Seismic attribute selection for unsupervised seismic facies analysis using user-guided data-adaptive weights: Geophysics, **83**, O31-O44. doi: 10.1190/GEO2017-0192.1

Figures

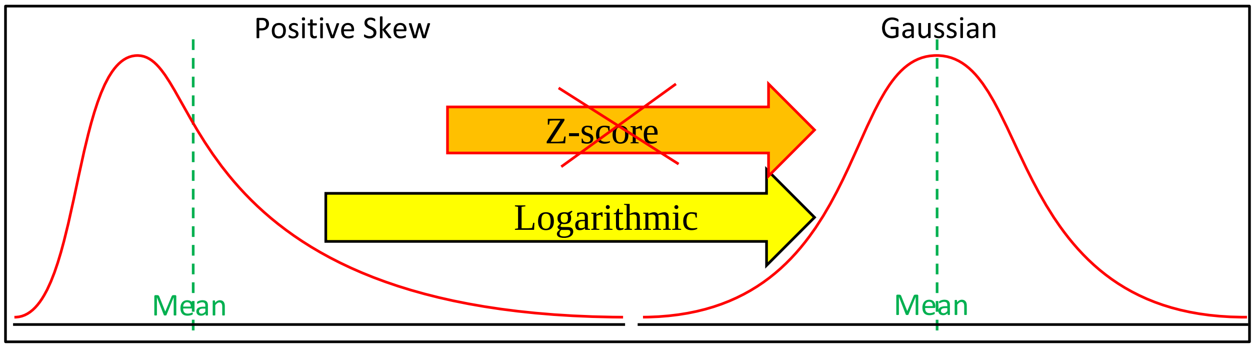


Figure 4.1. Most machine learning workflows assume the input data distributions are Gaussian. However, almost all seismic attributes have skewed distributions. Although conventional z-score normalization compensates for different data ranges, it does not reshape the data distribution. Can logarithmic normalization "Gaussianify" data distributions? How? And would logarithmic normalization give better machine learning results than the simple z-score normalization?

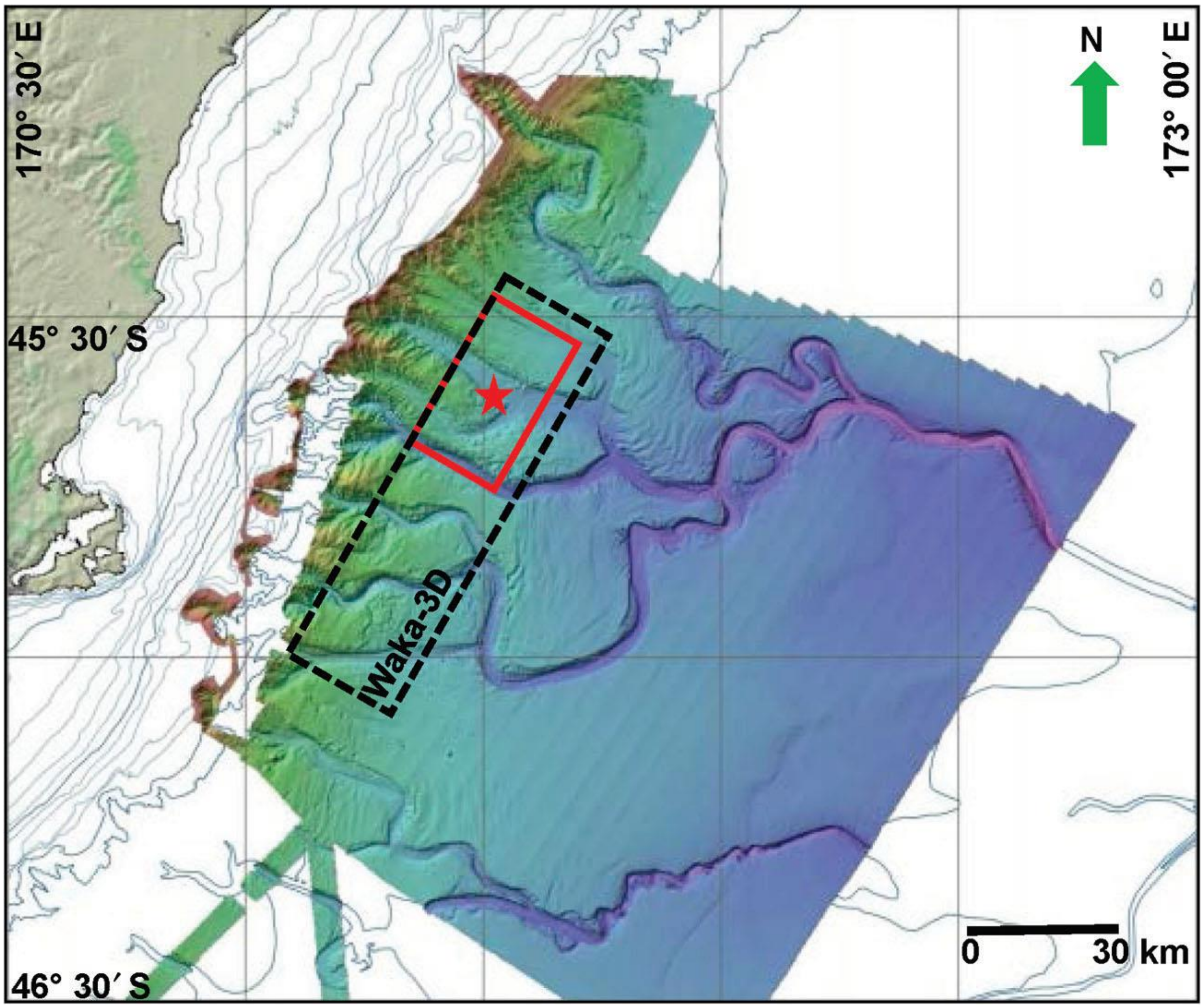


Figure 4.2. Seafloor bathymetry of the Canterbury Basin, New Zealand (Modified from Zhao et al., 2016). The study area indicated by the red rectangle is in the northern part of the Waka3D survey images multiple turbidites at the transition between continental shelf and slope.

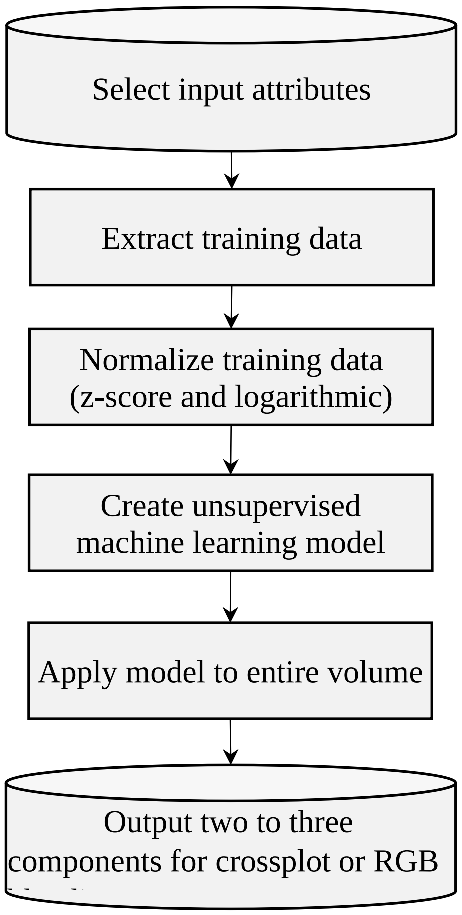

Select input attributes
Extract training data
Normalize training data (z-score and logarithmic)
Create unsupervised machine learning model
Apply model to entire volume
Output two to three components for crossplot or RGB

Figure 4.3. Flowchart of my unsupervised classification.

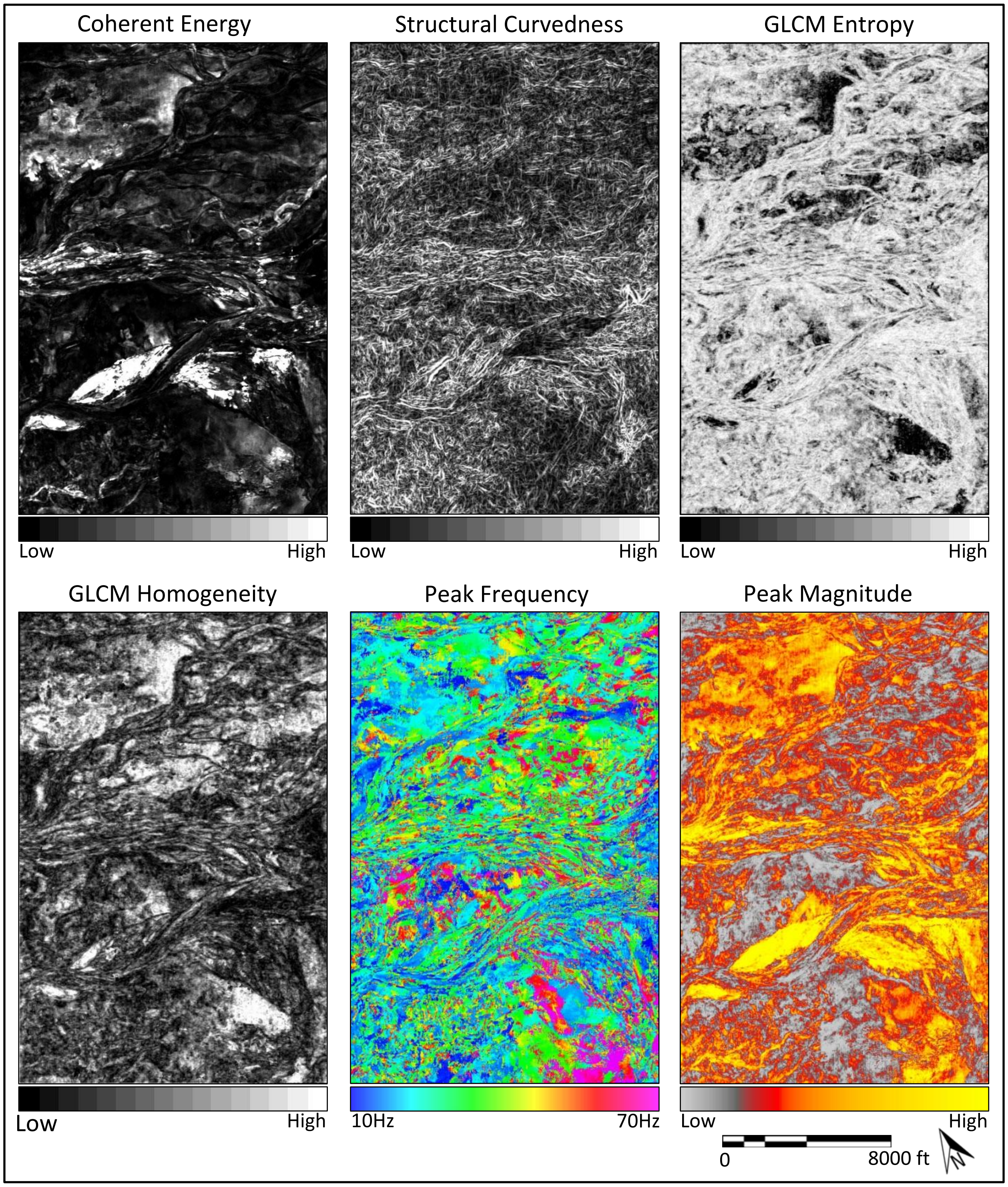


Figure 4.4. Horizon slices through six input seismic attributes used in unsupervised classification: coherent energy, structural curvedness, GLCM entropy, GLCM homogeneity, peak frequency, and peak magnitude. Training data were extracted from three adjacent horizon slices within the Late Tertiary sequences, showing various turbidite architectural elements.

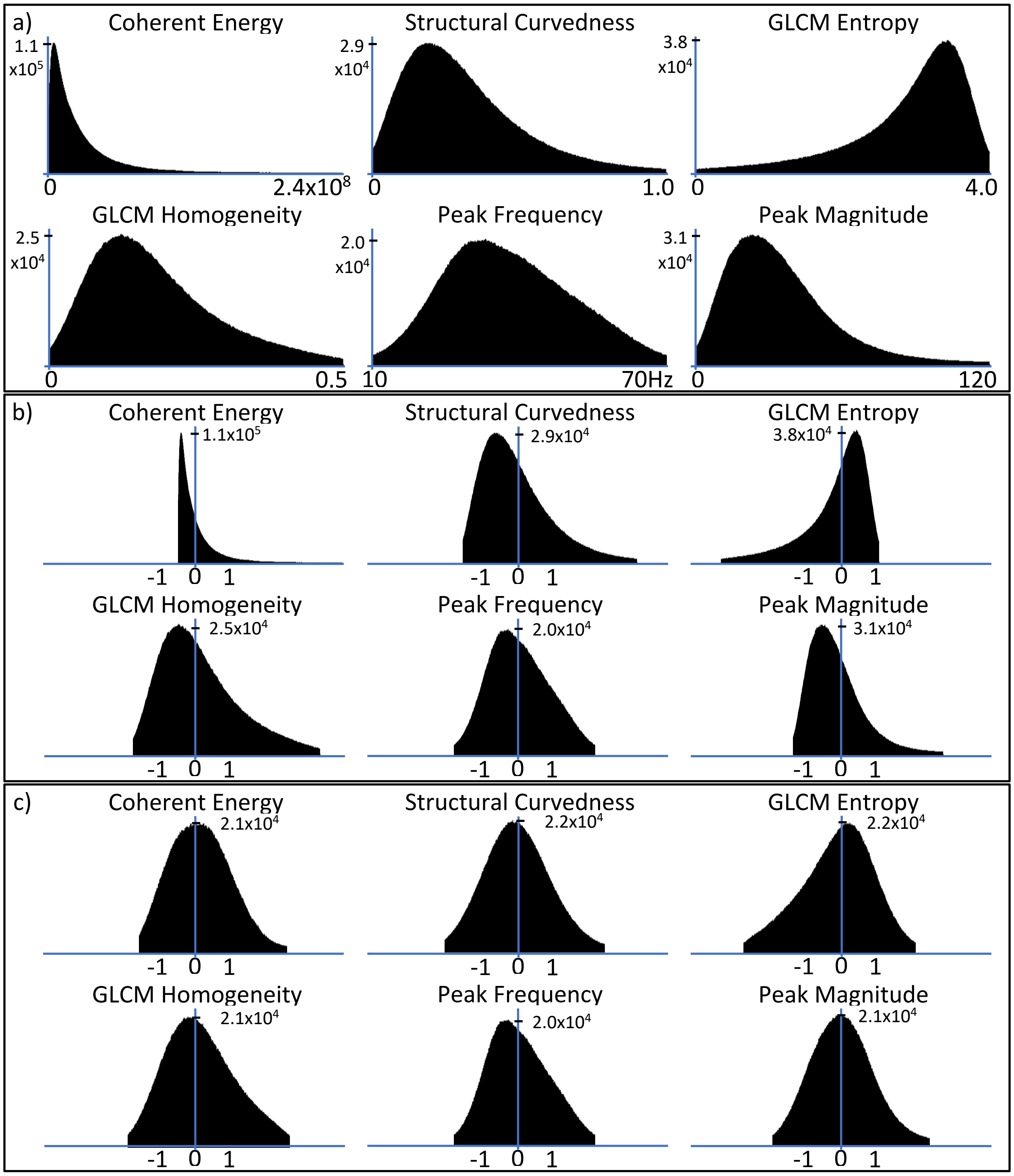


Figure 4.5. Histograms of (a) original data, (b) data after z-score normalization, and (c) data after logarithmic transformation. All inputs are non-negative and exhibit different value ranges. Z-score normalization shifts and stretches/squeezes the original distributions so that the resulted distributions have mean μ=0 and a standard deviation of σ=1. However, z-score normalization does not change the skewness of the original distributions. In contrast, a logarithmic transformation reshapes the distributions to better approximate a Gaussian distribution.

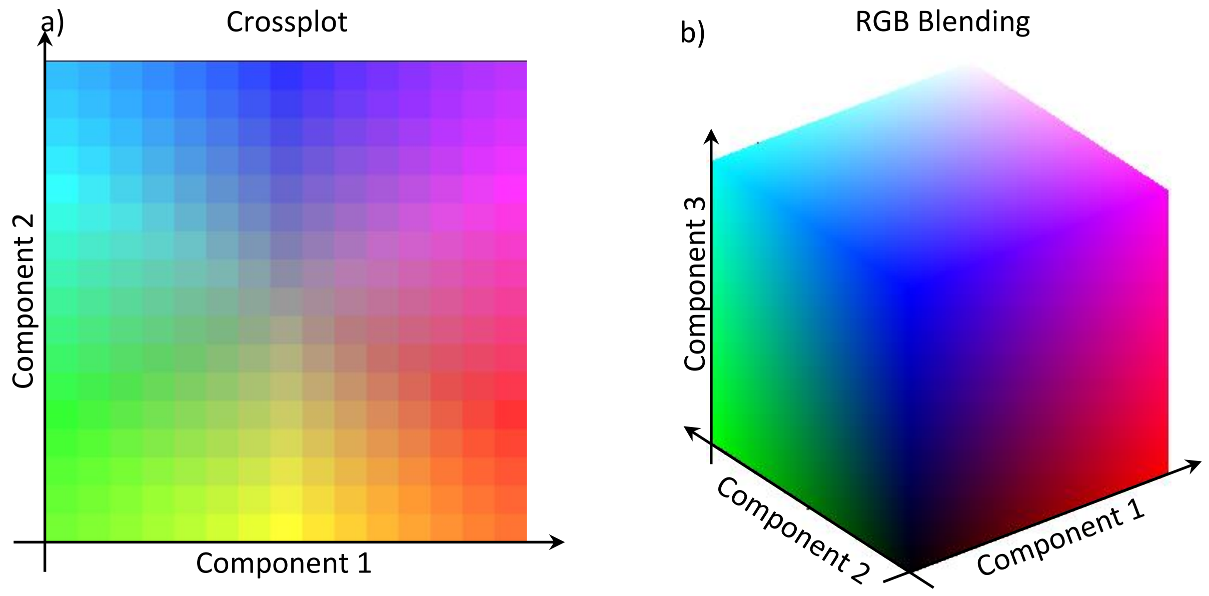


Figure 4.6. Color models used to (a) crossplot two components and (b) corender three components generated by unsupervised classification algorithms.

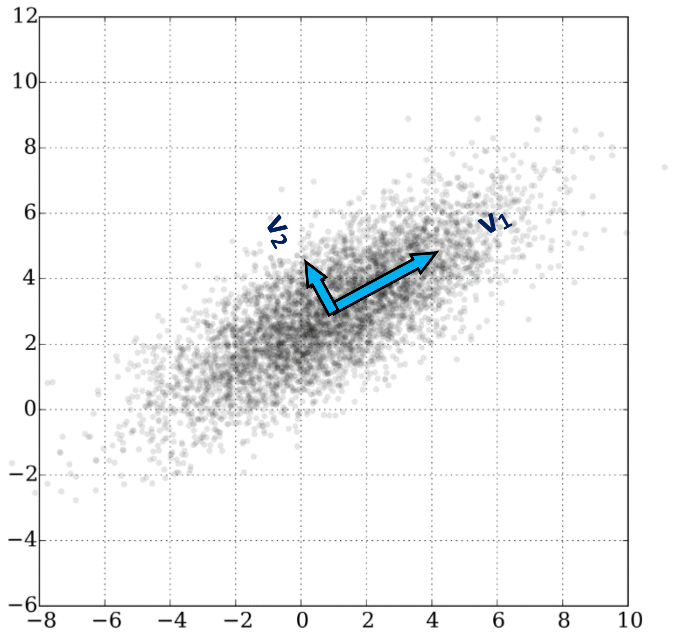


Figure 4.7. Illustration of Principal Component Analysis (PCA). The first eigenvector v_1 best represents the variation in data cloud. The first principal component (PCA 1) is generated by projecting each data point onto the v_1 axis. The second eigenvector v_2 best represents the data not represented by v_1 and is orthogonal to it. The second principal component (PCA 2) is generated by projecting each data point onto the v_2 axis.

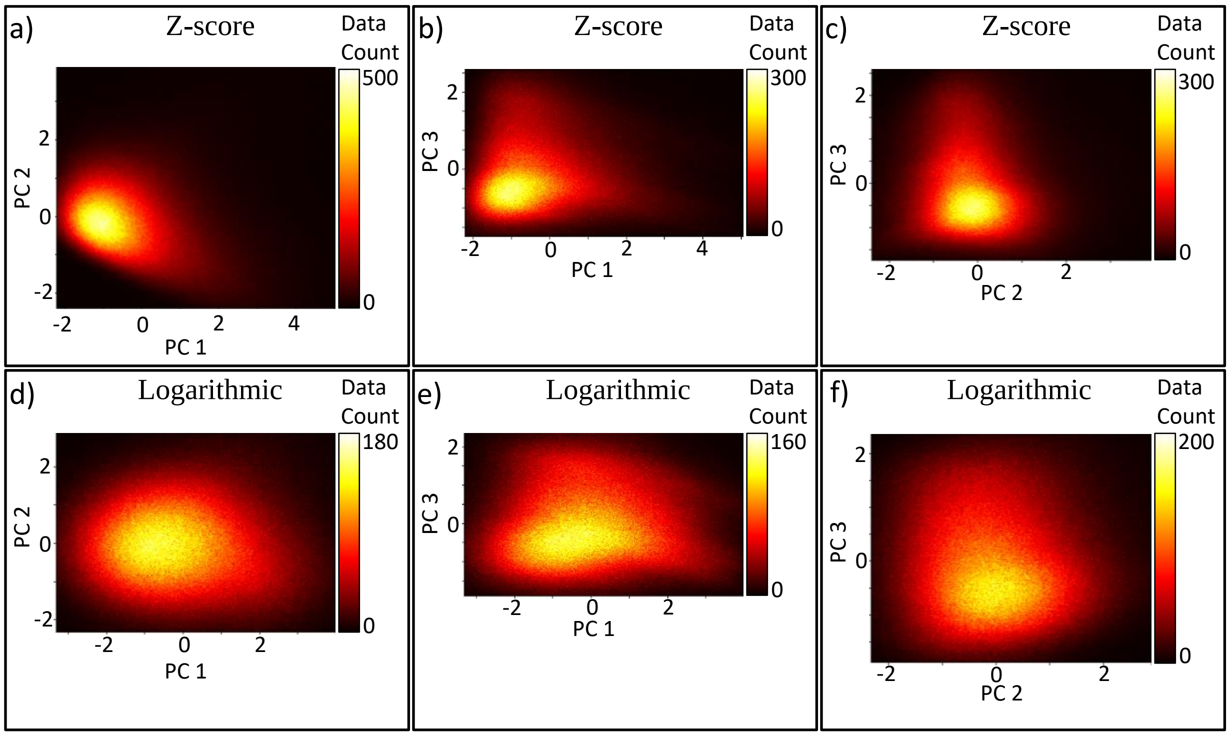


Figure 4.8. 2D histograms of the first three principal components computed from (a), (b), (c) z-score normalized and from (d), (e), (f) logarithmically transformed input. To avoid the effect of extreme values on the histograms, 5% data at the extreme positive and 5% data at the extreme negative of each principal component are clipped and fall outside the images. The logarithmic histograms exhibit more evenly distributed, elliptical, and symmetric distributions than the z-score histograms. Note that there is still a recognizable common pattern between each pair of histograms (a) to (d), (b) to (e), and (c) to (f). For example, (f) looks like a zoomed-in section near the center of (c).

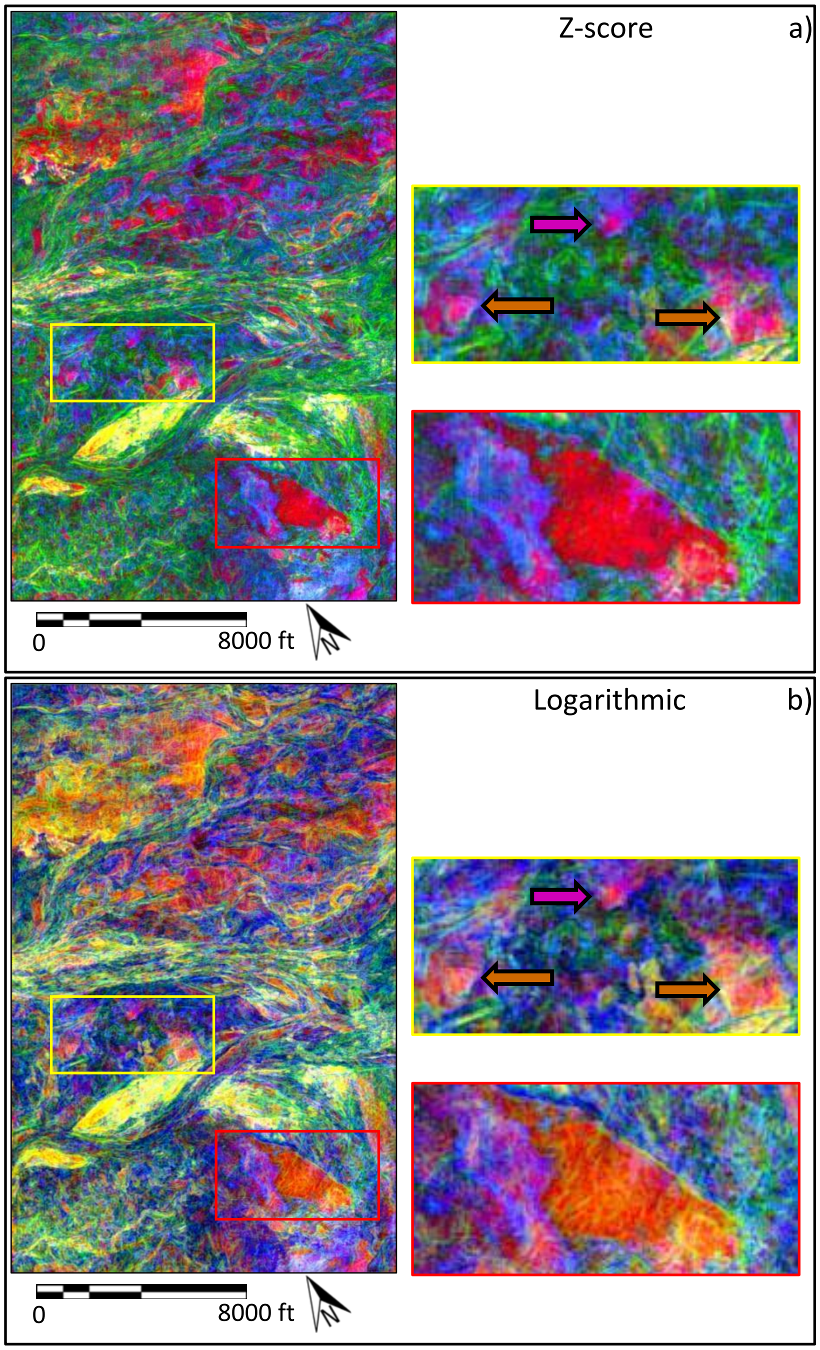
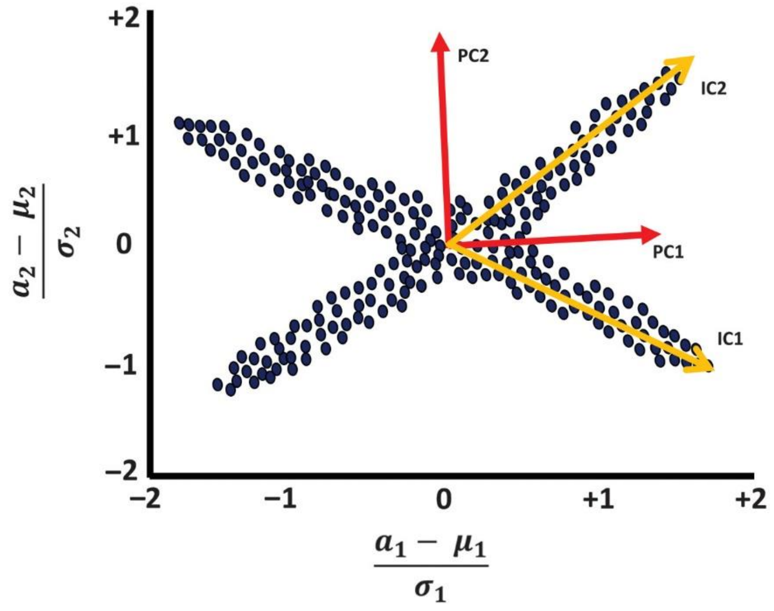


Figure 4.9. RGB blended images of three principal components computed using (a) z-score normalization and (b) logarithmic transformation. Overall, (b) is more evenly distributed against the 3D color bar whereas (a) is biased towards green. In the yellow box of (b), we can distinguish the crevasse splays indicated by the magenta and orange arrows from each other, whereas all three splays have the magenta purple color in (a). In the red box of (b), note the small, yellowish channels within the orange crevasse splay, whereas in (a), the splay appears as a homogeneous dark red mass.



+2
+1
0
−1
−2
−2
−1
0
+1
+2
PC2
PC1
IC2
IC1
$\frac{a_2 - \mu_2}{\sigma_2}$
$\frac{a_1 - \mu_1}{\sigma_1}$

Figure 4.10. Illustration of Independent Component Analysis (ICA) (Lubo-Robles and Marfurt, 2019). Instead of a single, multi-dimensional, ellipsoidal “cloud” of input data, there can be several “clouds” of different shape, size, and orientation in a data distribution, and thus an orthogonal coordinate system like principal components cannot fully capture the data variation. ICA fills this gap by using higher-order statistics to find a set of non-orthogonal independent components that better represent the non-orthogonal sub-distributions, thus better separating different geological features.

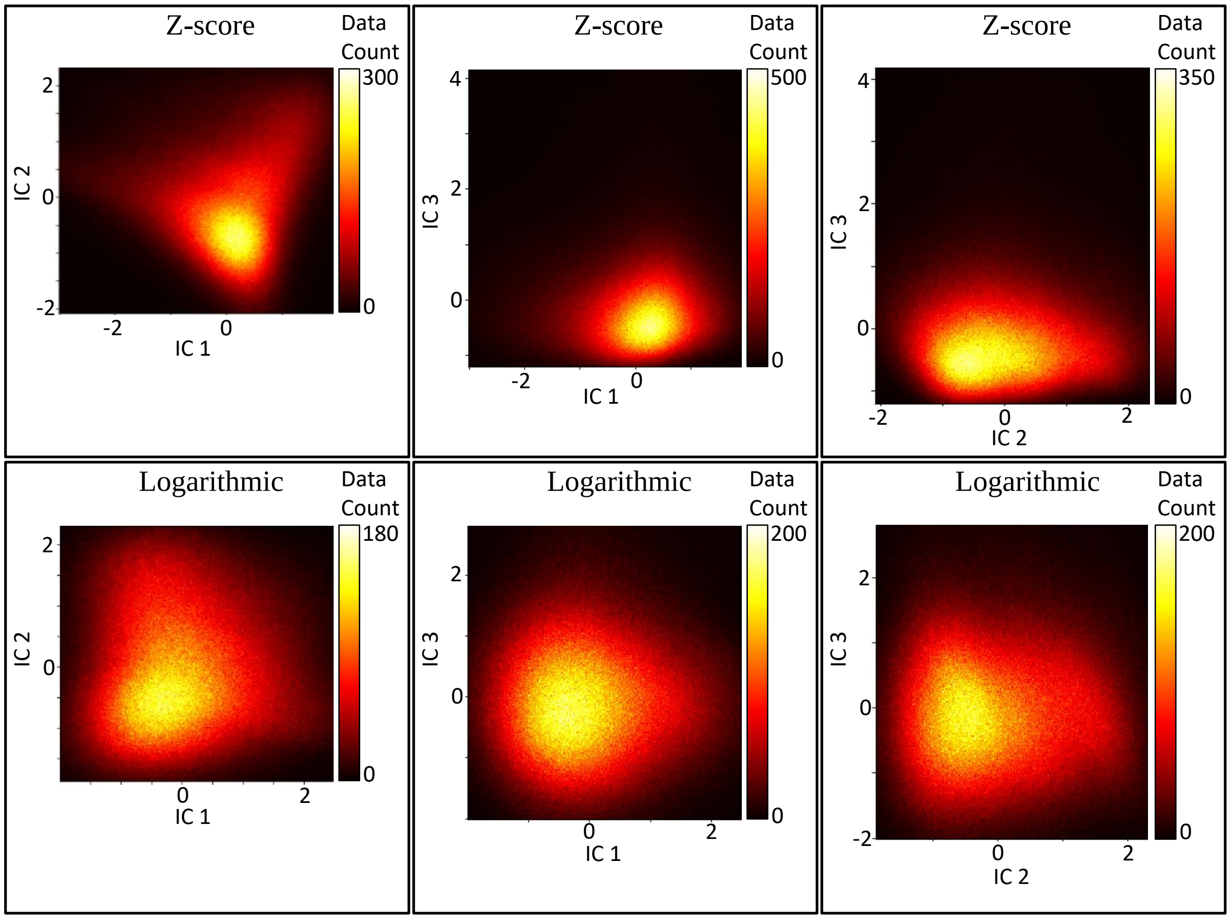


Figure 4.11. 2D histograms of independent components computed on z-score normalized inputs (a, b, c) and on logarithmically transformed inputs (d, e, f). Similar to PCA, 5% data at the extreme positive and 5% data at the extreme negative of each independent component are clipped. Again, logarithmic histograms show more spread-out, circular distributions than z-score histograms. Note that for each pair of independent components (a-d, b-e, c-f), z-score and logarithmic histograms show completely different clouds of data with no common pattern. This is likely because output independent components are not ranked, and using different normalization schemes somehow changes the polarity and/or the order of the output components.

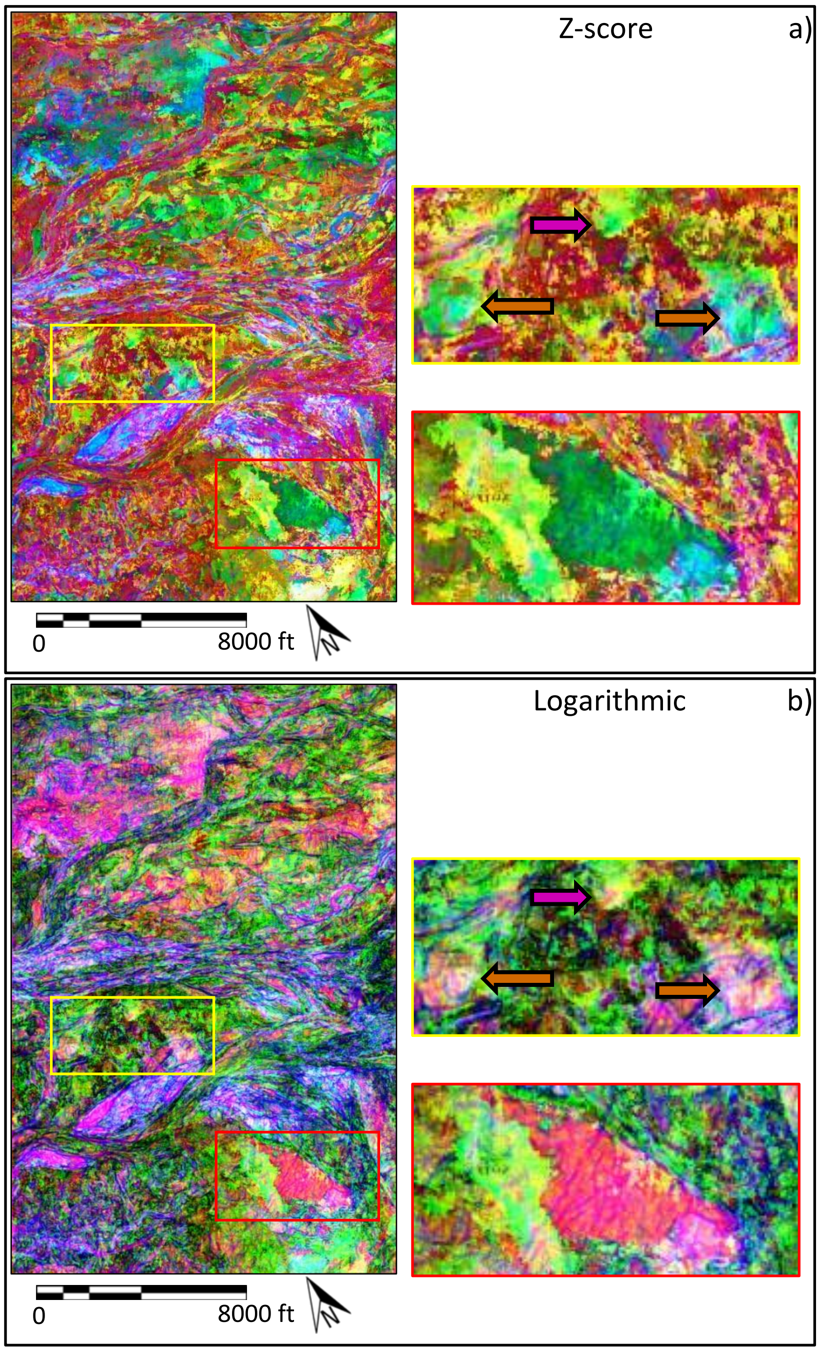


Figure 4.12. RGB blended images of three output independent components with (a) z-score normalization and (b) logarithmic transformation. (a) and (b) have completely different color gamut. In the yellow boxes, I can distinguish the upper crevasse splay (purple arrow) from the lower two crevasse splays (orange arrows) in both (a) and (b). Regarding the detail of a crevasse splay's internal structure (red boxes), both are on par with logarithmic PCA results in Figure 4.9b.

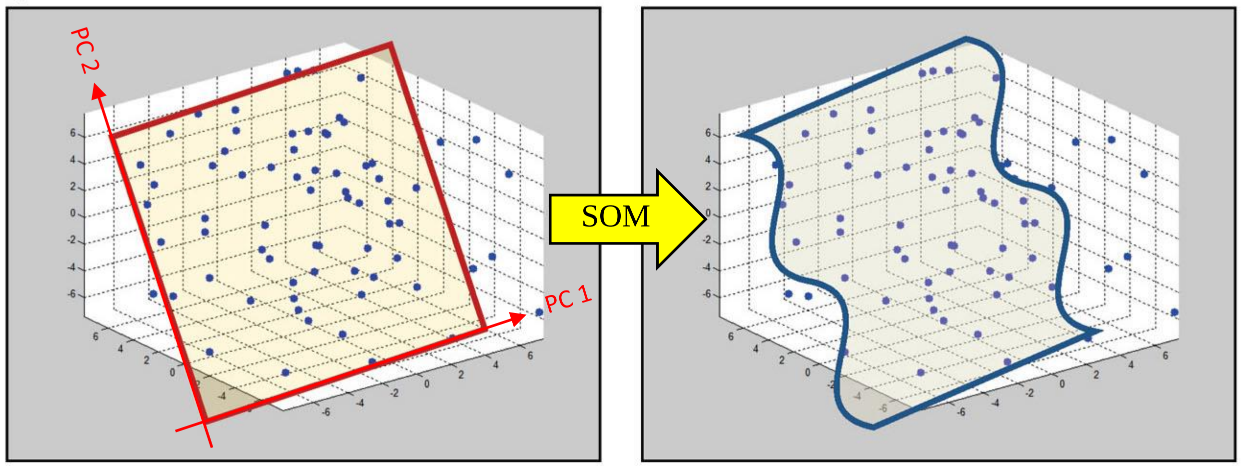


Figure 4.13. Illustration of Self-Organizing Maps (SOM) (modified from Zhao et al., 2015). SOM first initializes a 2D grid on the plane defined by the first two eigen vectors. This 2D grid is then deformed iteratively into a 2D manifold that best fit the input training data. Clustering is done by “snapping” a data sample to the nearest grid node.

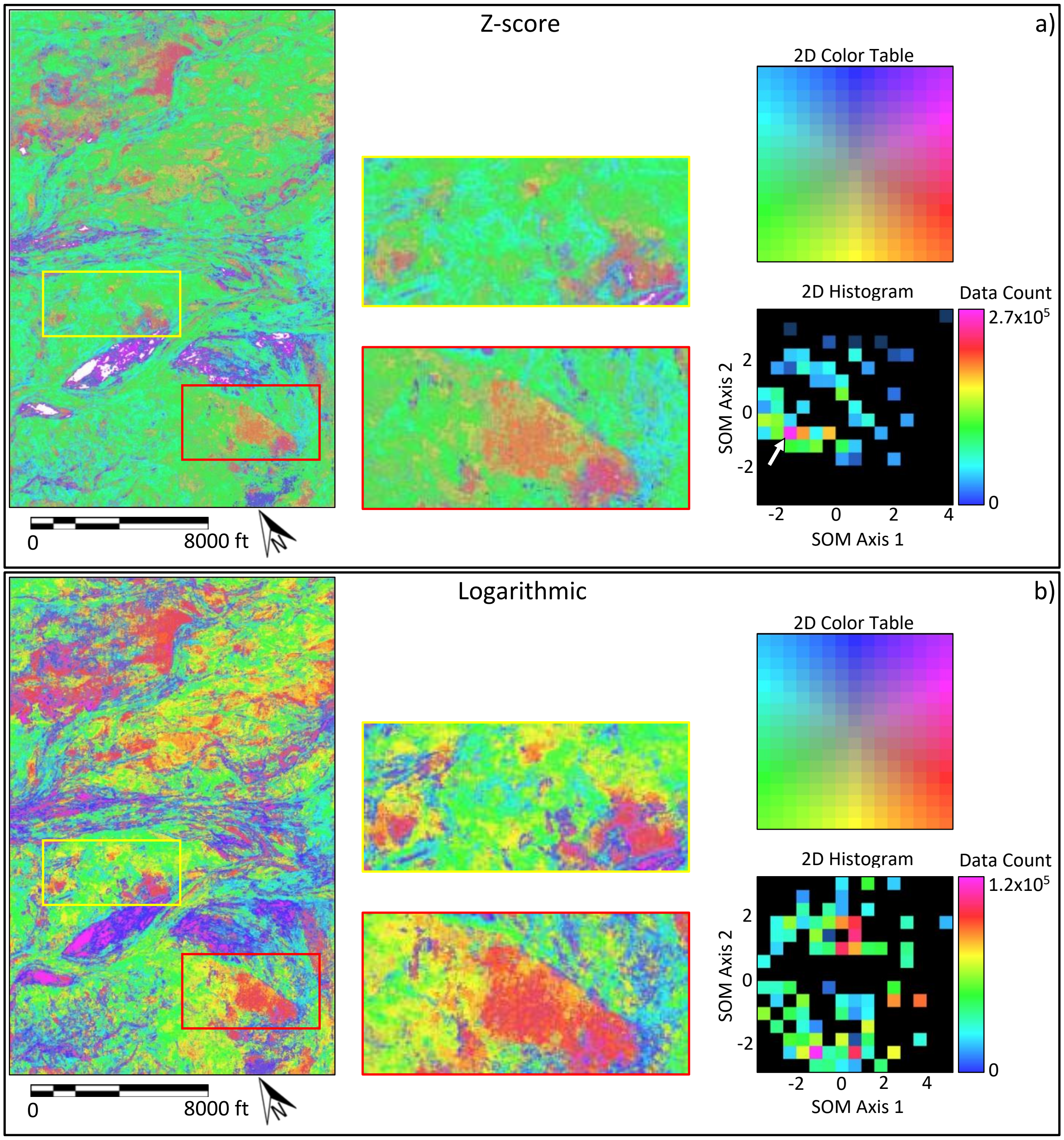


Figure 4.14. 2D color crossplot of two output SOM components with (a) z-score normalization and (b) logarithmic transformation. I clip the extreme values of the z-score SOM components to allows the z-score image in (a) to have approximately the same level of color contrast as the logarithm scaled image (b), which does not require any data clipping at all. White pixels in (a) represent clipped data. Even after such strong clipping, the z-score image is biased towards green representing the data distribution skewed to the lower left of the 2D histogram. Many of the data points are highly concentrated within one cell (white arrow). The logarithm transformation image in (b) is more diffuse and better spans the 2D color bar, thereby better delineating the yellow and orange crevasse splays from the green flood plain.

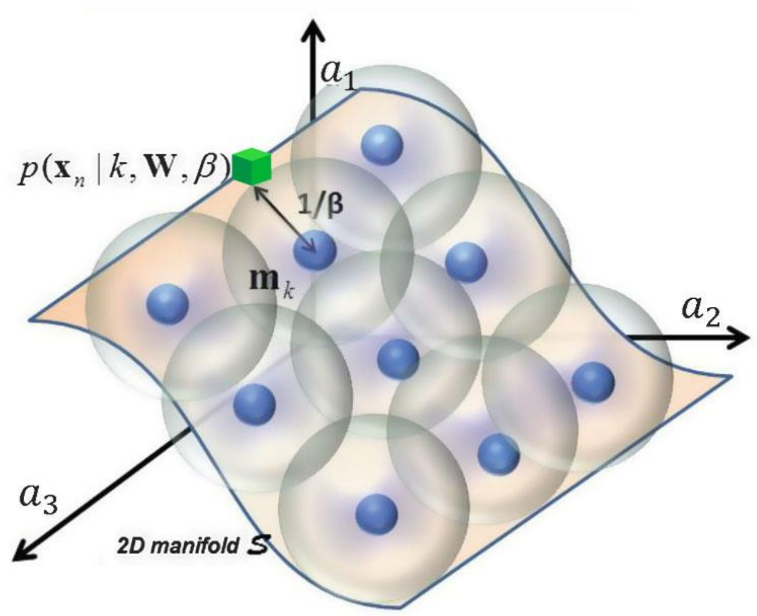


Figure 4.15. Illustration of Generative Topographic Mapping (GTM) (modified from Zhao et al., 2015). Similar to SOM, GTM also generates a 2D gridded manifold that best fits the input training data. However, instead of “snapping” a data sample (green cube) to the closest grid node, each node (blue spheres) is given a Gaussian distribution function (big translucent grayish spheres) to measure the probability of a data sample to a grid node. All Gaussian distributions are of the same size. This size and the positions of the Gaussian distributions change with each iteration in the clustering process. At the end, the “responsibility” of each node to a given data vector is computed. In unsupervised classification, the most “responsible” grid node of a data sample is plotted against a 2D color table. In semi-supervised classification (e.g. Roy et al., 2014; Qi et al., 2016) a probability density function is computed and output as a separate volume for each class, resulting in a measure of probability that a data sample belongs to a given class.

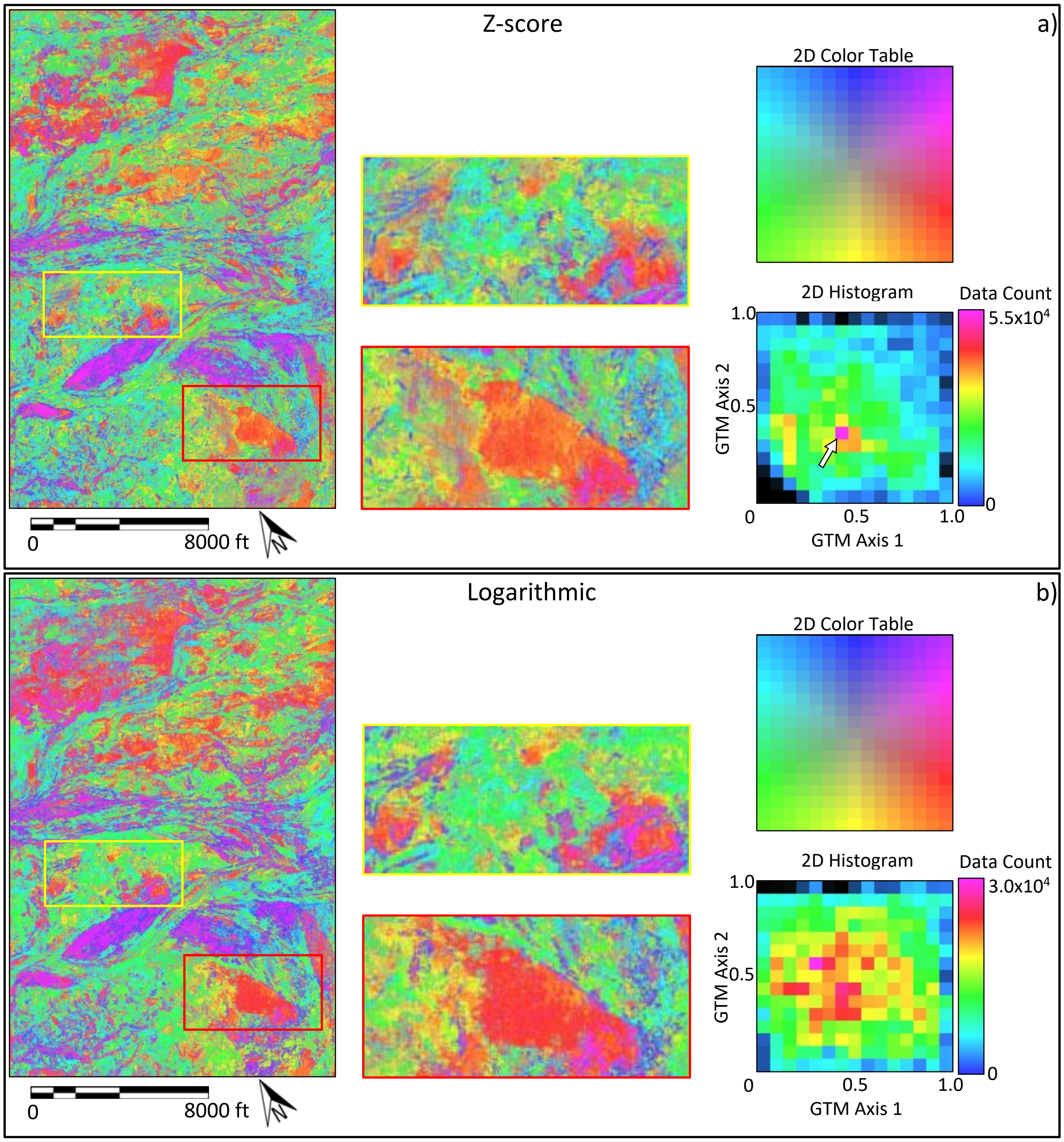


Figure 4.16. 2D color crossplot of two output GTM components with (a) z-score normalization and (b) logarithmic transformation. No data clipping is needed in both (a) and (b). Again, data are highly concentrated at a cell just below the center of the z-score 2D histogram, whereas logarithmic 2D histogram shows a more spread-out distribution. The two crossplots have very similar color gamut, and at first glance, the z-score crossplot appears to better enhance smaller "features". However, a close inspection reveals that these "features" are not geologically meaningful, but rather represent random noise, causing the real geological features like channels and crevasse splays to be less continuous than those in the logarithmic crossplot.

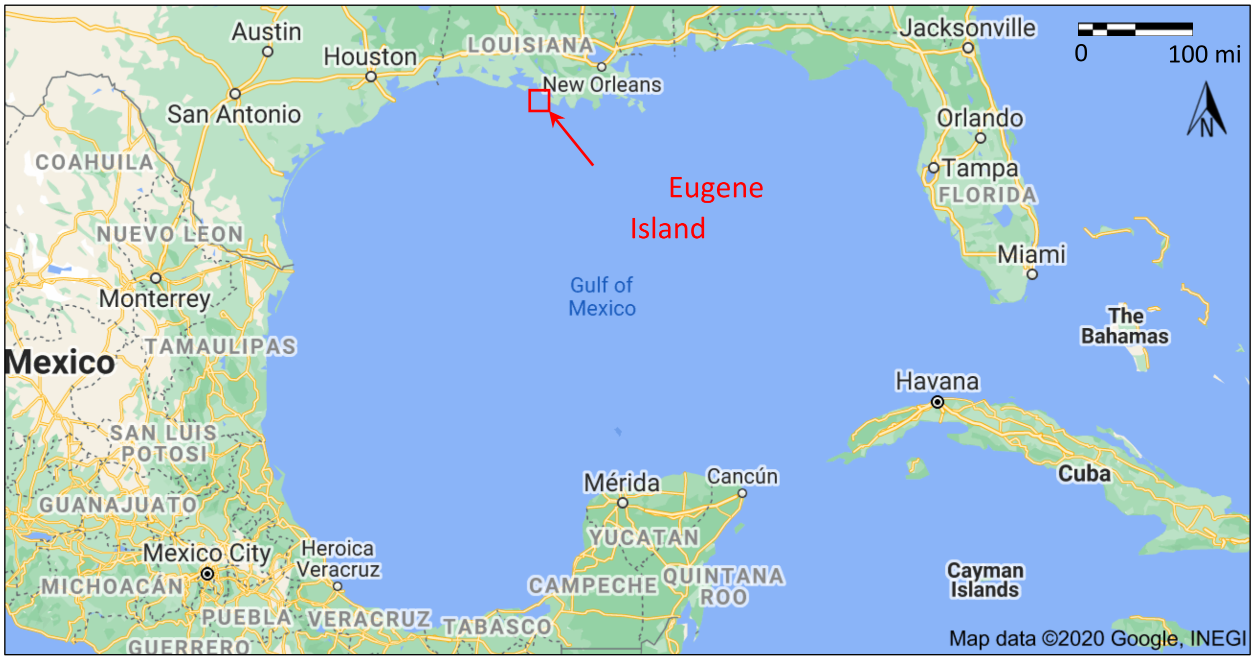


Figure 4.17. Eugene Island mini-basin, Gulf of Mexico.

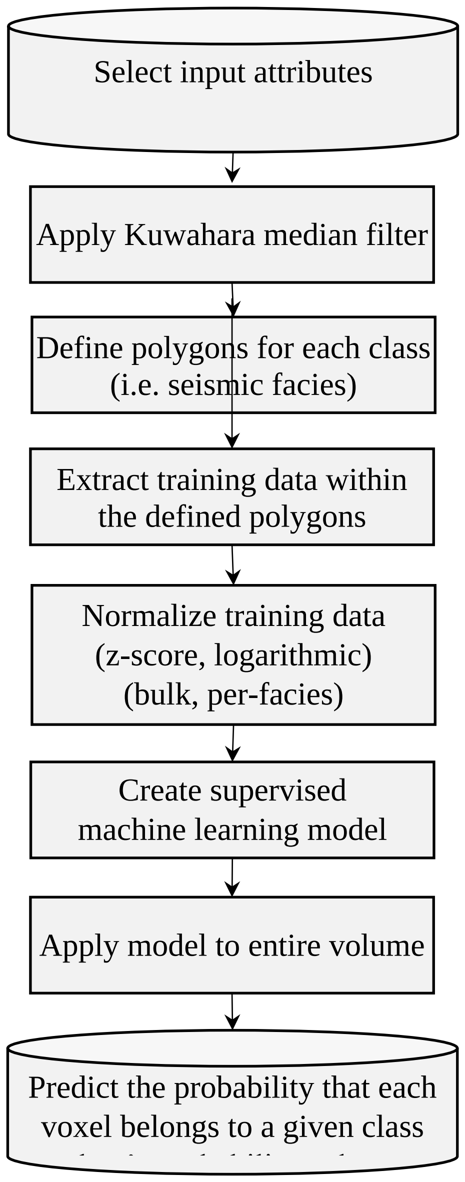
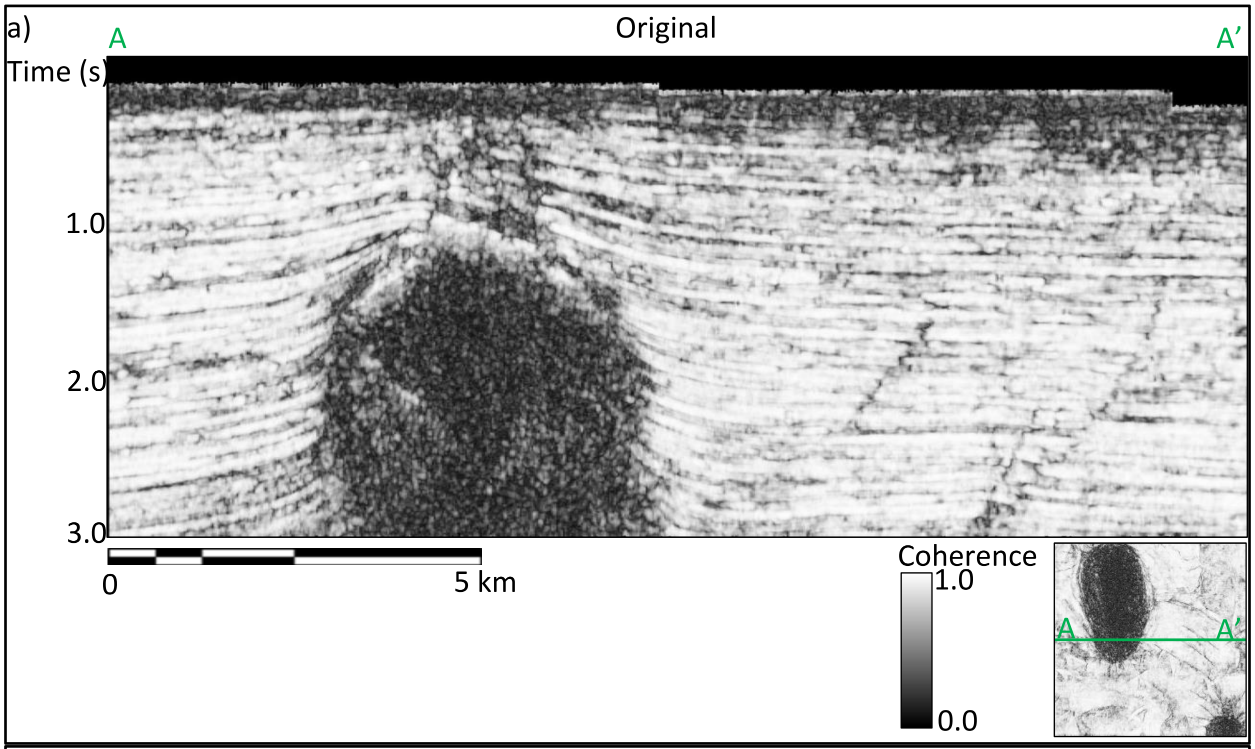

Select input attributes
Apply Kuwahara median filter
Define polygons for each class
(i.e. seismic facies)
Extract training data within
the defined polygons
Normalize training data
(z-score, logarithmic)
(bulk, per-facies)
Create supervised
machine learning model
Apply model to entire volume
Predict the probability that each
voxel belongs to a given class

Figure 4.18. Flowchart of my PNN supervised classification workflow.



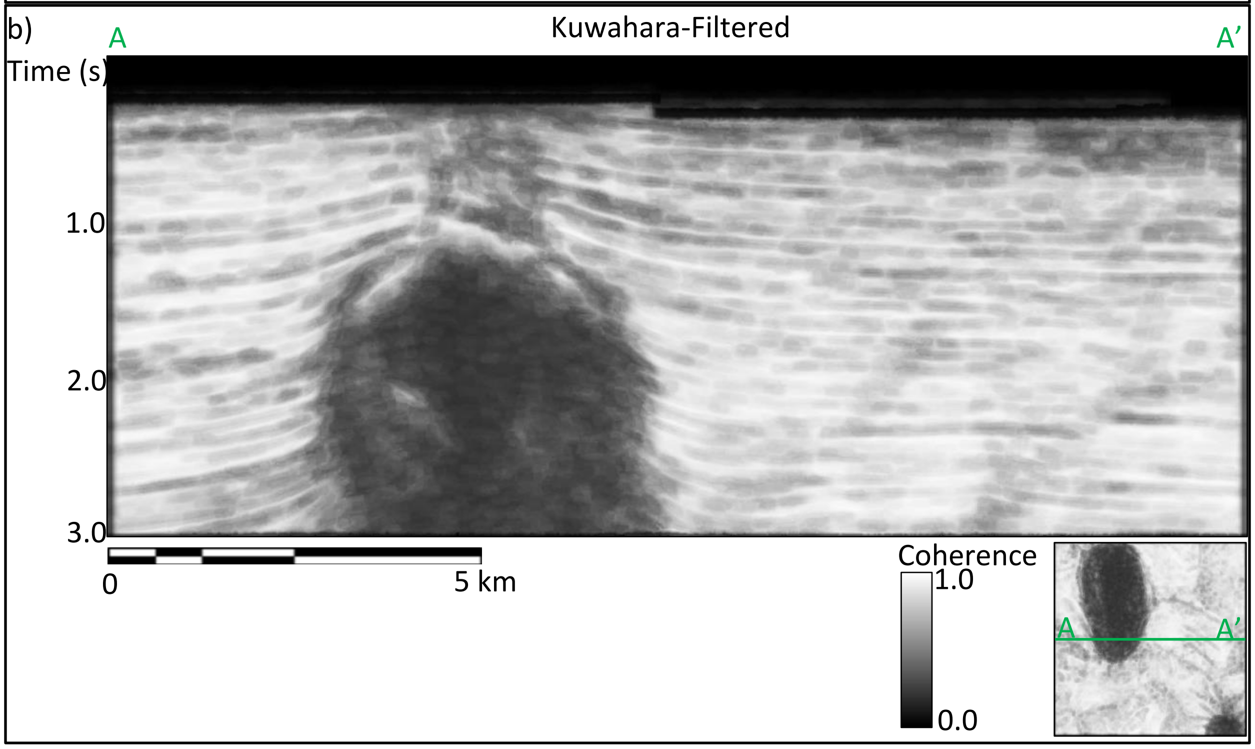


Figure 4.19. AA' Vertical slices through the (a) original and (b) Kuwahara-filtered coherence volumes. The Kuwahara filter smooths the internal detail of a salt diapir and sharpens the edges. Because the low coherence faults are thinner than the 3×3×3 Kuwahara window, they are also attenuated, avoiding their misclassification as a seismic facies.

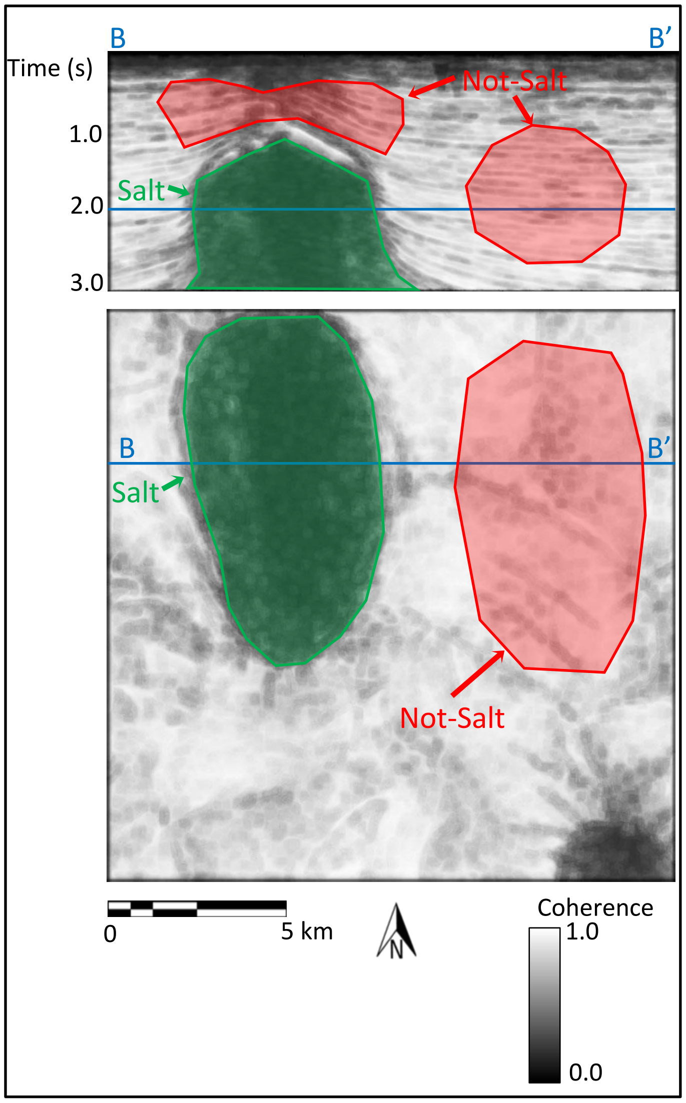


Figure 4.20. Vertical slice BB’ and time slice at *t*=2.0s through the Kuwahara-filtered coherence volume, showing the picked polygons of salt and not-salt facies. I carefully define the polygons in such a way that the area covered by the salt polygons is approximately equal to the area covered by the not-salt polygons. I then extract supervised training data from these polygons.

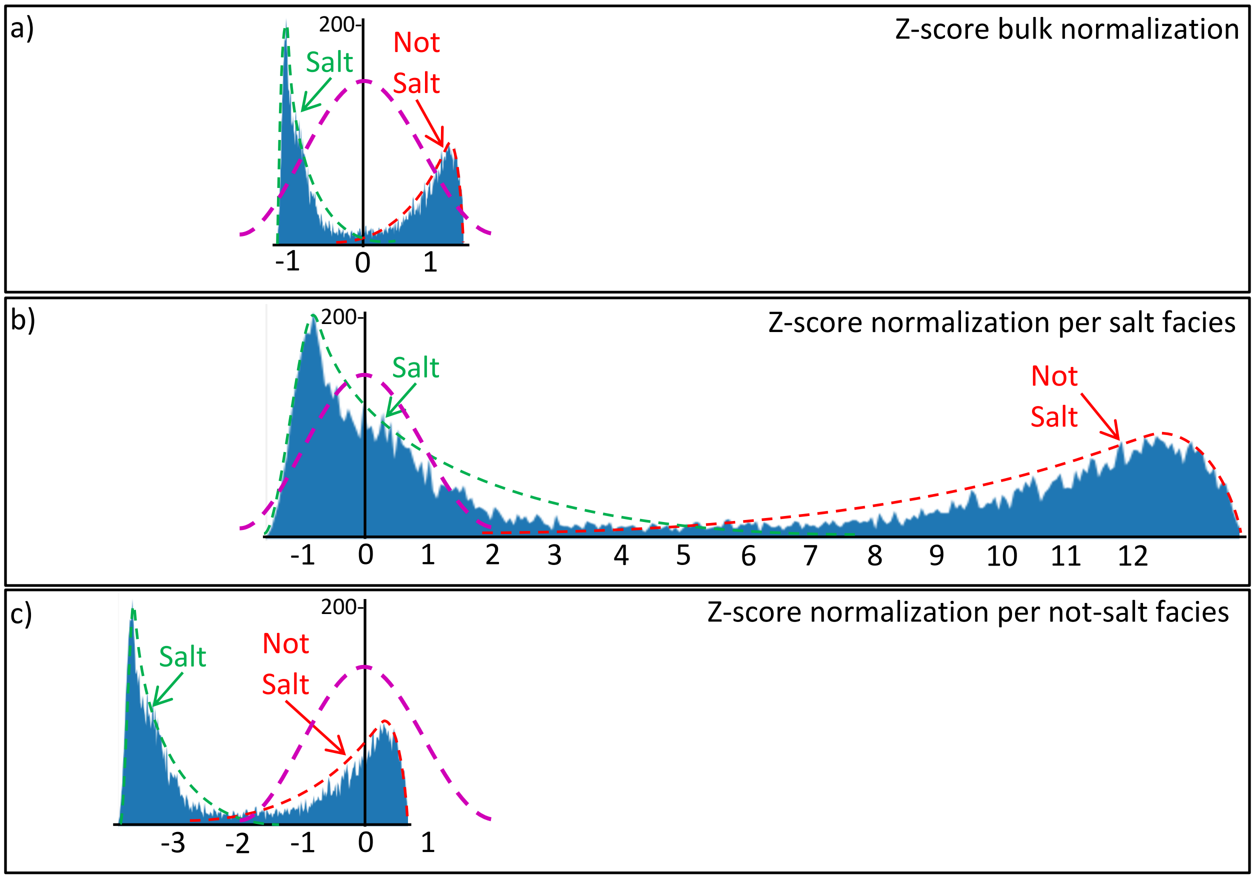


Figure 4.21. Difference between bulk normalization (a), in which I apply the same normalization to the training data of all classes, and per-class normalization (b and c), in which I normalize the training data of each class separately. The histograms were aligned at zero and resized to have the same horizontal scale. Magenta dashed lines represent ideal normal distribution curves – the ultimate desired outcome of normalization. With bulk normalization, the ideal curve is in the middle of the two clusters, while with per-class normalization, the ideal curve is at each cluster's distribution, though not perfectly aligned.

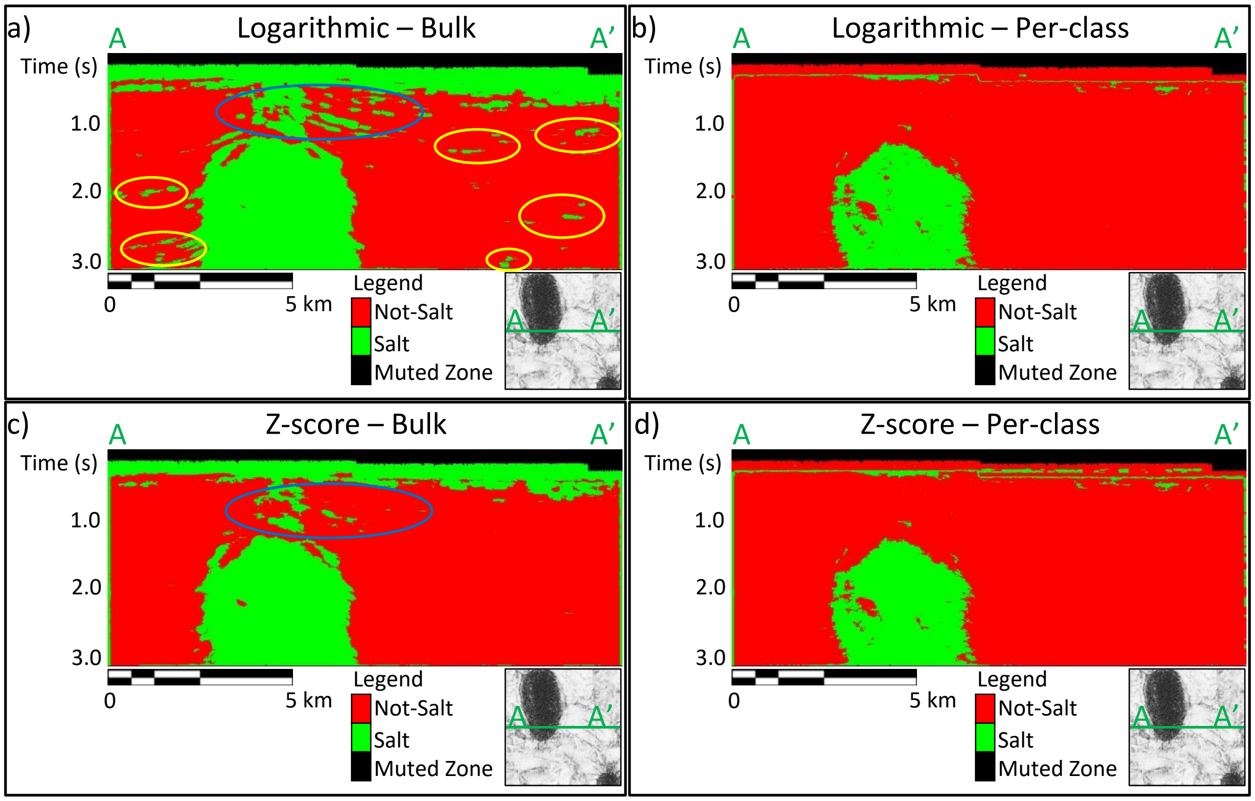


Figure 4.22. Vertical slices AA' through the PNN prediction volumes associated with (a) logarithmic bulk normalization, (b) logarithmic per-class normalization, (c) z-score bulk normalization, and (d) z-score per-class normalization. Yellow ellipses highlight some mis-classified salt patches within the not-salt region of (a). Blue ellipses highlight the faulted region above the salt diapir that are mis-classified as salt in (a) and (c). Logarithmic per-class normalization (b) and z-score per-class normalization (d) produces very similar predictions.

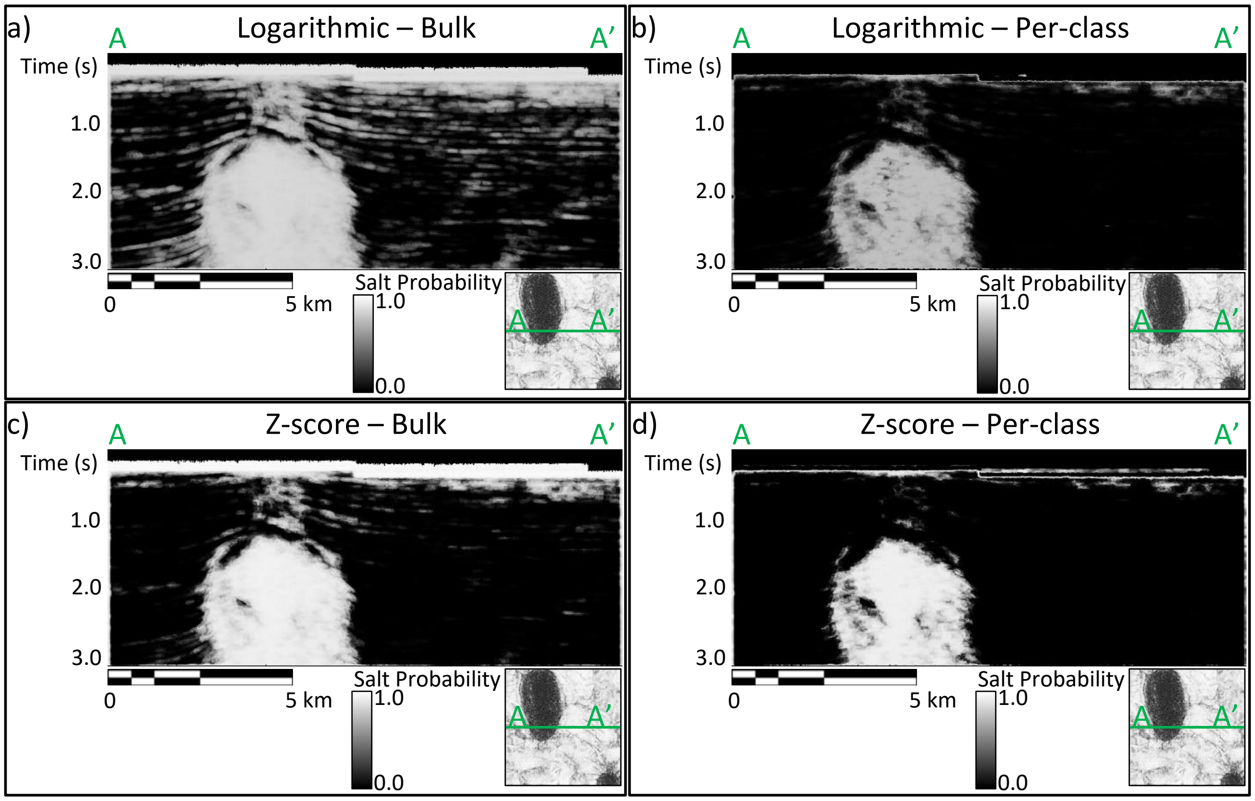


Figure 4.23. Vertical slices AA' through the PNN salt probability volumes associated with (a) logarithmic bulk normalization, (b) logarithmic per-class normalization, (c) z-score bulk normalization, and (d) z-score per-class normalization. All displays have the same fixed scaling of salt probability from 0.0 to 1.0. Note the dimmed image produced by logarithmic per-facies normalization (b), while z-score per-facies normalization image (d) has a much higher contrast.

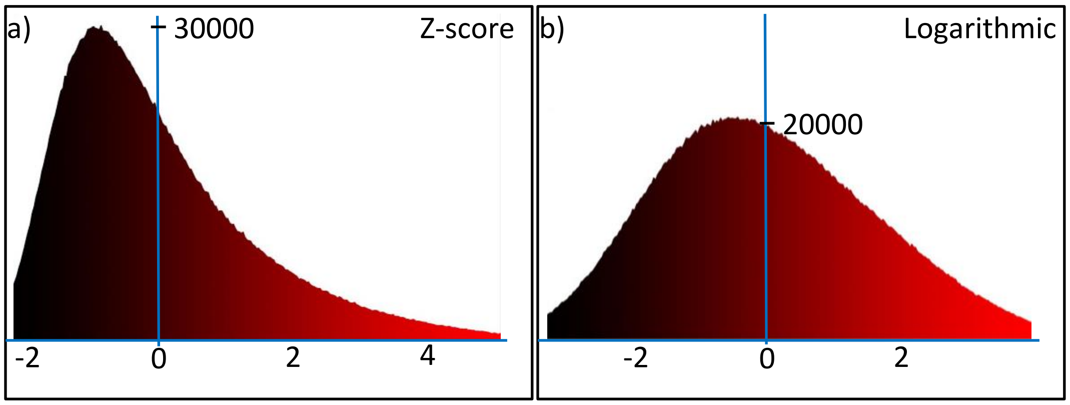


Figure 4.24. Histograms of the first principal component projection associated with (a) z-score normalization and (b) logarithmic transformation, blended with black-to-red color gradient to illustrate the process of color mapping against the red color channel. The histograms are clipped in such a way that only 90% of the data distribution is shown, while 5% percentile in the extreme left and 5% percentile in the extreme right of the distribution are discarded. Note the wider and more symmetric histogram curve associated with logarithmic transformation, allowing more colors to be mapped near the peak of the data distribution, making the final RGB blended image of principal component projections capable of showing subtle, fine detail of the turbidite channel system in Canterbury Basin, New Zealand.

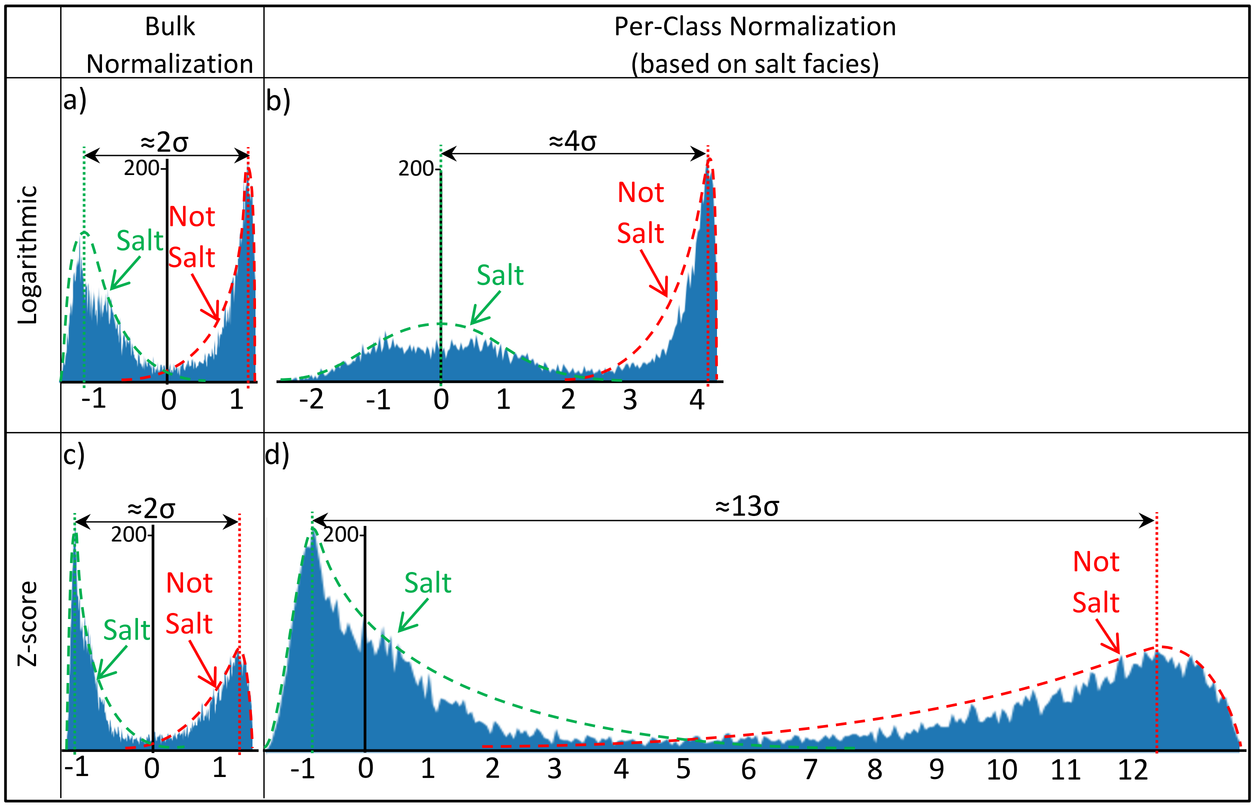


Figure 4.25. Histograms of PNN's input coherence attribute, showing different shapes of clusters and distances between clusters via different normalization schemes: (a) logarithmic bulk transformation, (b) logarithmic transformation based on training data of salt facies, (c) z-score bulk normalization, and (d) z-score normalization based on training data of salt facies. Among the histograms, z-score per-class normalization distribution (d) has the largest distance between salt and not-salt clusters, which corresponds to the PNN salt probability image with the highest contrast (Figure 4.23d). The general trend is that z-score histograms have greater distances between clusters than logarithmic histograms, and per-class normalization scheme yields significantly greater distances between clusters than bulk normalization scheme. Note that in (a) and (c), the distances between the two clusters are approximately the same, but salt cluster in logarithmic histogram (a) has a wider distribution than z-score histogram (c), causing many coherent data points to be mis-classified as salt (yellow ellipses in Figure 4.22a).

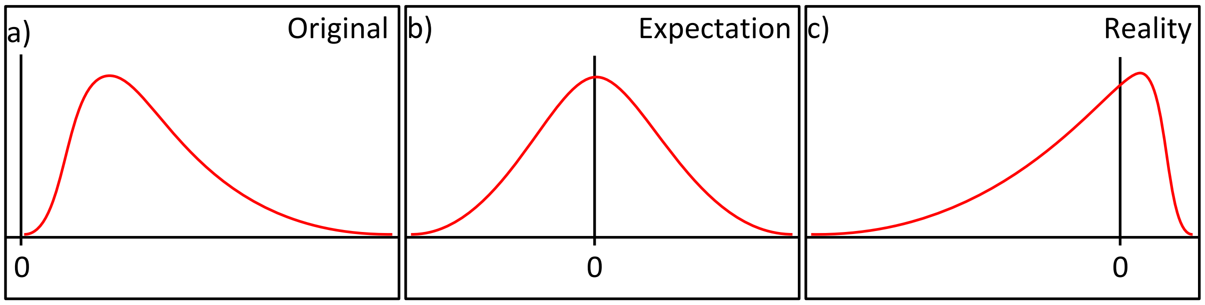


Figure 4.A-1. Illustrations of (a) original distribution of a typical non-negative, right-skewed attribute, (b) the expected symmetric, “bell”-shaped distribution after logarithmic transformation, and (c) the bitter reality of directly applying a logarithmic function to the original data: a left-skewed, even more asymmetric than original distribution with a left “tail” stretching to negative infinity.

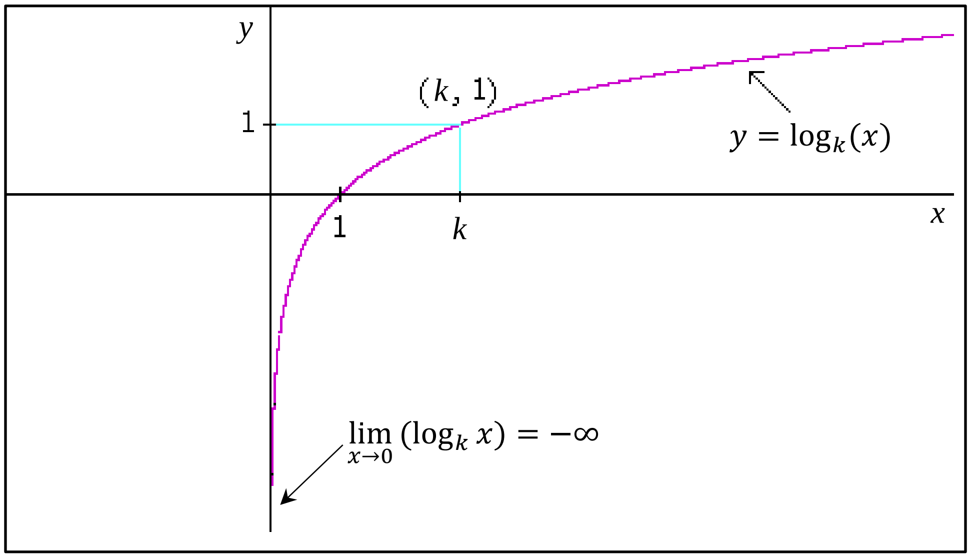


Figure 4.A-2. Graph of the logarithmic function. As *x* goes to zero, the logarithmic function approaches negative infinity, thus causing the long left “tail” of the actual distribution in Figure 4.A-1c. Also, note that logarithmic function is not defined where *x*<0.

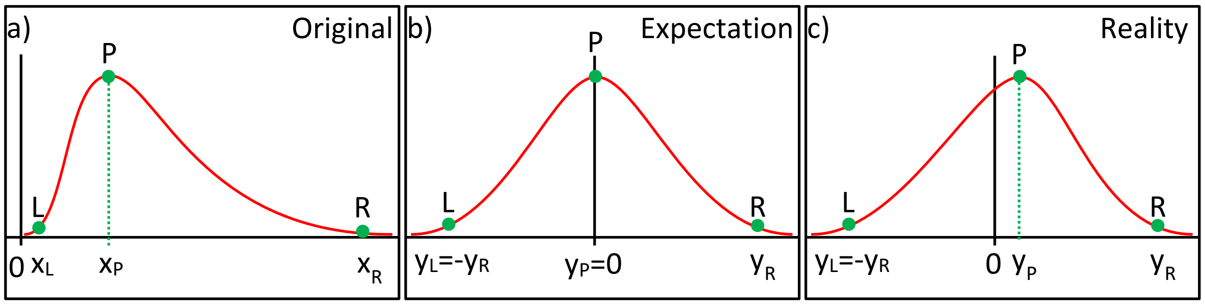


Figure 4.A-3. Three anchor points on the original distribution (a): left (L), right (R), and peak (P). The goal of the logarithmic transformation is to reshape the distribution in such a way that the left and right anchor points are symmetric about zero, while the peak is exactly at zero (b). However, even after a careful derivation of logarithmic parameters using the three-anchor point scheme, the reality (c) is still far from expectation because the logarithmic transformation moves the relative location of the peak! This means I need to compute the parameters of the logarithmic transformation in an iterative manner.

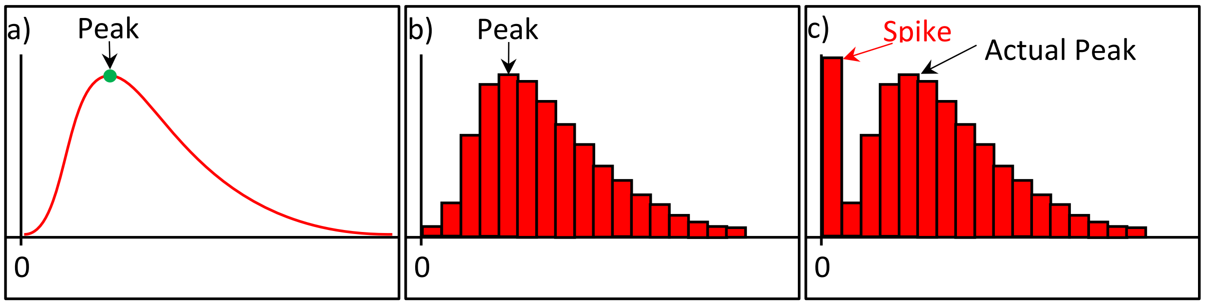


Figure 4.B-1. (a) Illustration of the peak (or statistical mode) of a distribution. The traditional procedure to find the peak is to construct histogram columns and locate the column with the greatest number of points (b). The precision of the peak is determined by the width of a histogram column. However, if there is a spike in the distribution, such as zero-value samples belonging to dead traces and muted zones, the highest histogram column could represent the spike instead of the actual peak (c).

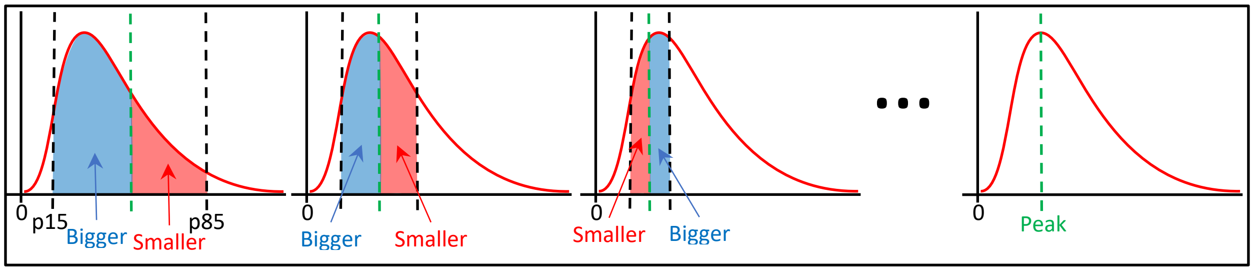


Figure 4.B-2. Illustration of my method to find the peak of a data distribution using a binary search algorithm. First, I limit the search between the 15th and 85th percentiles of the distribution to avoid the effect of spikes at the extreme left and right, assuming the peak is most likely located within this range near the distribution center. I divide this range into two halves of the same width and count the data points residing in each half. I then choose the half with greater data count, divide it into another two halves and start counting data points within these new halves again. The process is repeated until there is exactly one data point in each half, at which time I define the peak of the distribution as the average of the last two data points.

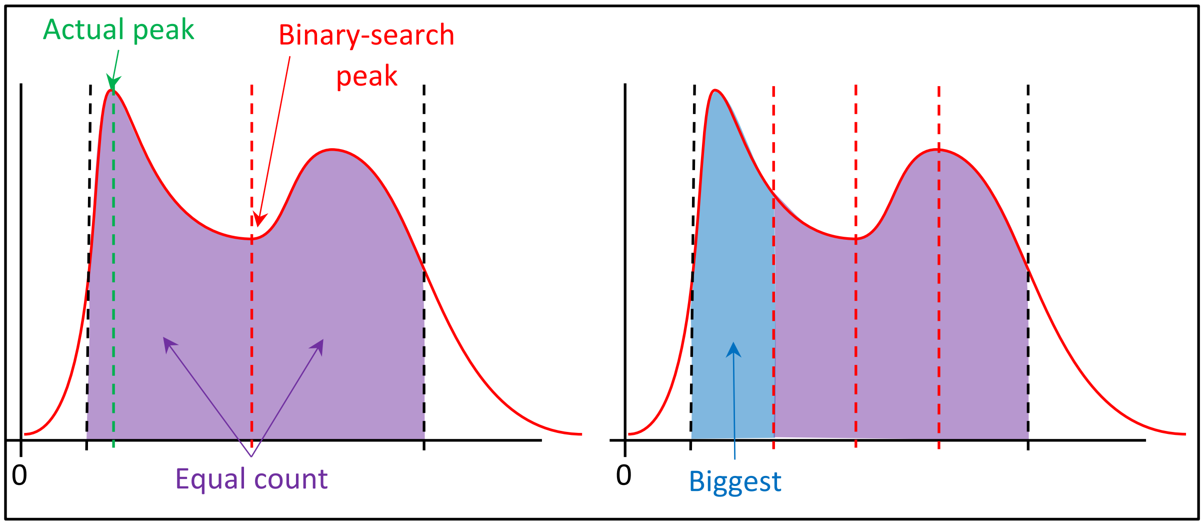


Figure 4.B-3. In some rare occurrences, when a data distribution has more than one cluster and the total number of data points is relatively small, it is possible that after a division, the two halves have exactly the same data count of a large quantity. If this happens, the binary search would stop, and the midpoint of the two halves is considered to be the peak, while the actual peak may reside in one of the two halves. A partial solution to this issue is to further divide two halves into four quarters and find the quarter with the maximum data count, assuming the peak belongs to the densest quarter.

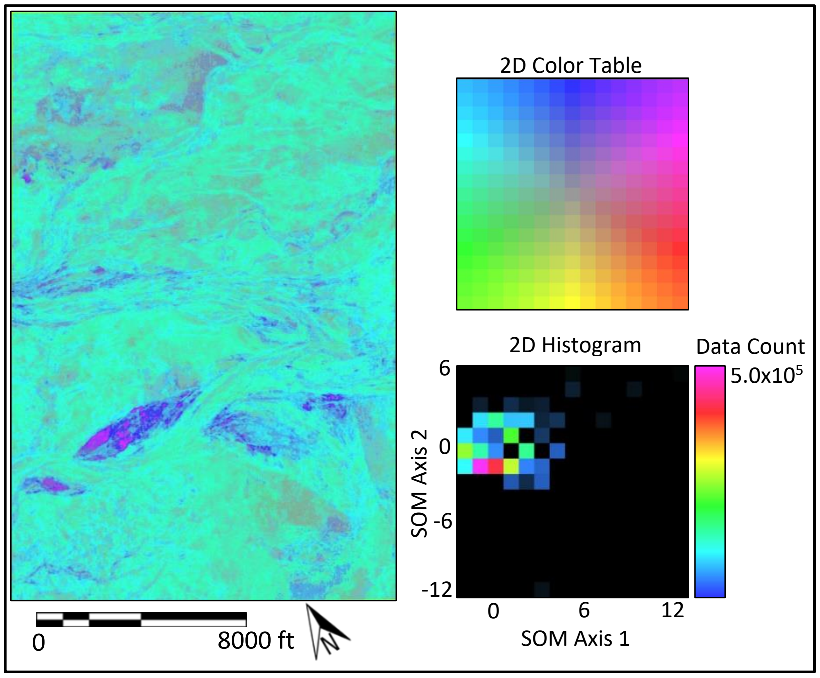


Figure 4.C-1. 2D color crossplot of two output SOM components using z-score normalization, without any data clipping. The crossplot is mostly cyan because the data distribution is heavily skewed to the upper left corner of the 2D histogram, which corresponds to a variety of cyan colors in the 2D color table. I can barely see any geological detail due to low color contrast.

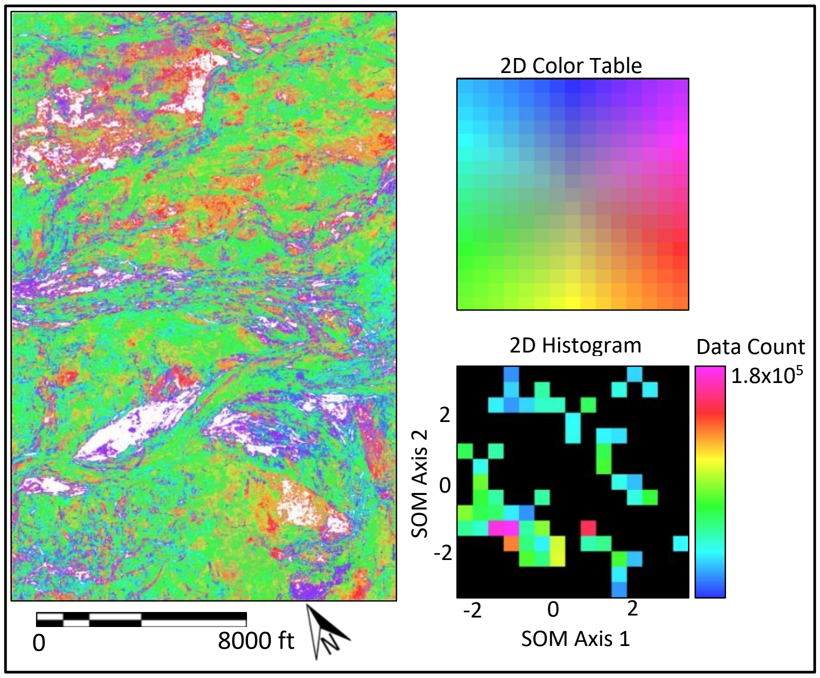


Figure 4.C-2. 2D color crossplot of two output SOM components using z-score normalization, with strong data clipping. The color contrast is good, but the crossplot has too many white pixels, which represent clipped data points.

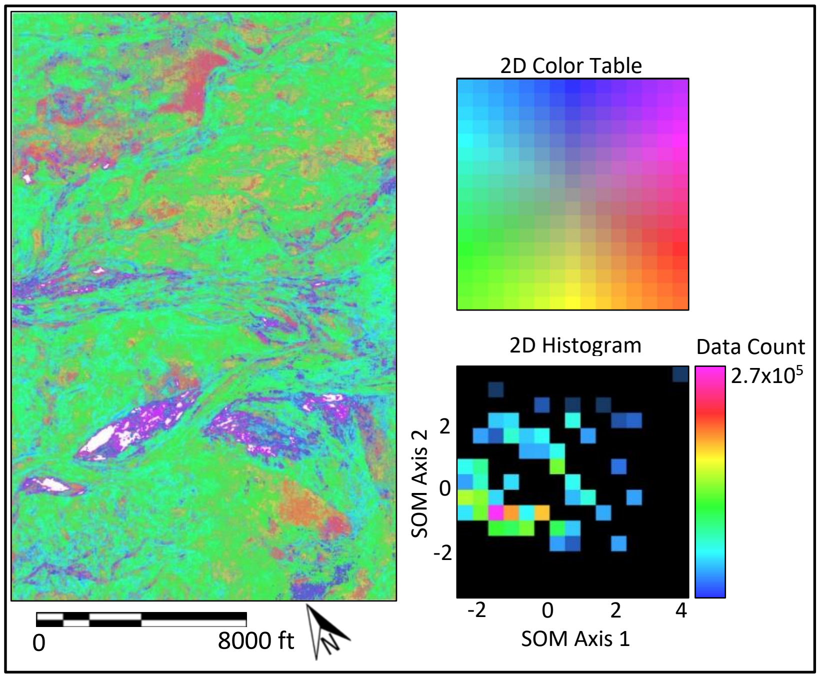


Figure 4.C-3. 2D color crossplot of two output SOM components using z-score normalization, with moderate data clipping. The number of white pixels (i.e. clipped data points) are greatly reduced, and the color contrast increased over Figure C-1. This figure is the one displayed as Figure 4.14a.

Tables

Table 4.D-1. Optimal combination of input attributes of each normalization scheme, together with the corresponding PNN smoothing parameter r and validation error. Note that coherence attribute is found in all four normalization schemes' optimal combination of input attributes.

Normalization Scheme	**Optimal Combination of Input Attributes**	**PNN smoothing parameter *r***	**Validation Error (lower is better)**
Logarithmic bulk	Coherence, GLCM Contrast, GLCM Entropy	2.1	0.047
Logarithmic per-facies	Coherence, GLCM Contrast	0.6	0.031
Z-score bulk	Coherence	1.3	0.037
Z-score per-facies	Coherence, Most Positive Curvature	1.9	0.024

Table 4.E-1. Parameters of dip computation in case study 1.

Parameter	**Value**
Algorithm	Semblance Search
Maximum angle searched (degree)	15
Search angle increment (degree)	3
Time-to-depth conversion velocity (m/s)	4000
Vertical window half height (s)	0.02
Inline window radius (m)	12.5
Crossline window radius (m)	25.0

Table 4.E-2. Parameters of dip filtering in case study 1.

Parameter	**Value**
Algorithm	LUM
Lower-Upper-Median percentile	20
Vertical window half height (s)	0.02
Inline window radius (m)	12.5
Crossline window radius (m)	25.0

Table 4.E-3. Parameters of coherence computation in case study 1.

Parameter	**Value**
Vertical window half height (s)	0.02
Inline window radius (m)	12.5
Crossline window radius (m)	25.0
Similarity Power	2.0
Low cut filter rolloff $f_{\mathbf{low}}$ (Hz)	5
High cut filter rolloff $f_{\mathbf{high}}$ (Hz)	100

Table 4.E-4. Parameters of curvature computation in case study 1.

Parameter	**Value**
Curvature type	Structural
Filter corner point λ_1 (m)	22016.2
Filter corner point λ_2 (m)	800
Filter corner point λ_3 (m)	400
Filter corner point λ_4 (m)	200
Filter weight w_1	1
Filter weight w_2	0.666
Filter weight w_3	0.333
Filter weight w_4	0
Constant multiplier of curvature	1000
Maximum operator radius (m)	1000
Vertical compression factor	0.25
Operator truncation value	0.01

Table 4.E-5. Parameters of GLCM computation in case study 1.

Parameter	**Value**
Vertical window half height (s)	0.004
Inline window radius (m)	25.0
Crossline window radius (m)	50.0
Number of gray levels	33

Table 4.E-6. Parameters of spectral decomposition in case study 1.

Parameter	**Value**
Spectral balancing factor (%)	4
Bluing exponent	0
Line and CDP decimation	5
Ormsby filter corner point $\boldsymbol{f}_1$ (Hz)	5
Ormsby filter corner point $\boldsymbol{f}_2$ (Hz)	10
Ormsby filter corner point $\boldsymbol{f}_3$ (Hz)	100
Ormsby filter corner point $\boldsymbol{f}_4$ (Hz)	120
CWT mother wavelet bandwidth (Hz)	0.26051
Temporal taper (s)	0.1
Percentile excluded in spectral shape	0.15
Lowest output frequency $f_{\mathbf{low}}$ (Hz)	10
Highest output frequency $f_{\mathbf{high}}$ (Hz)	100
Output frequency increment $\Delta \boldsymbol{f}$ (Hz)	1

Table 4.E-7. Parameters of training data extraction in case study 1.

Parameter	**Value**
Inline Start	1005
Inline End	1570
Inline Increment	1
Crossline Start	4645
Crossline End	6395
Crossline Increment	1
Vertical Boundary Type	About a horizon
Horizon	Late Tertiary
Window above horizon (s)	0.004
Window below horizon (s)	0.004
Vertical Increment (s)	0.004

Table 4.E-8. Parameters of PCA in case study 1.

Parameter	**Value**
Number of principal components	3
Minimum display percentile (%)	5
Maximum display percentile (%)	95

Table 4.E-9. Parameters of ICA in case study 1.

Parameter	**Value**
Number of independent components	3
Error Tolerance	1.0E-6
Maximum number of iterations	500
Minimum display percentile (%)	5
Maximum display percentile (%)	95

Table 4.E-10. Parameters of SOM in case study 1.

Parameter	**Value**
Number of prototype vectors (maximum number of classes)	256
Number of standard deviations along the first two eigenvector directions to define the initial 16*16=256 prototype vectors on the manifold	±4
Initial Neighborhood Scale	1.2
Distance type	Mahalanobis
Maximum number of training iterations	50
Grid spacing	150
Z-score clipping range of SOM axis 1	-2.6 to 4.0
Z-score clipping range of SOM axis 2	-3.5 to 3.5

Table 4.E-11. Parameters of GTM in case study 1.

Parameter	**Value**
Number of samples in 2D latent space	256
Number of basis functions	144
Relative width of basis functions	0.5
Weight regularization factor	0.05
Number of training iterations	50

Table 4.E-12. Parameters of dip computation in case study 2.

Parameter	**Value**
Algorithm	GST
Time-to-depth conversion velocity (ft/s)	10000
Vertical window half height (s)	0.02
Inline window radius (ft)	82.5
Crossline window radius (ft)	82.5

Table 4.E-13. Parameters of dip filtering in case study 2.

Parameter	**Value**
Algorithm	LUM
Lower-Upper-Median percentile	20
Vertical window half height (s)	0.02
Inline window radius (ft)	82.5
Crossline window radius (ft)	82.5

Table 4.E-14. Parameters of coherence computation in case study 2.

Parameter	**Value**
Vertical window half height (s)	0.02
Inline window radius (ft)	82.5
Crossline window radius (ft)	82.5
Similarity Power	2.0
Low cut filter rolloff $f_{\mathbf{low}}$ (Hz)	5
High cut filter rolloff $f_{\mathbf{high}}$ (Hz)	100

Table 4.E-15. Parameters of median smoothing in case study 2.

Parameter	**Value**
Number of iterations	5
Vertical window half height (s)	0.008
Inline window radius (ft)	165.0
Crossline window radius (ft)	165.0

Table 4.E-16. Parameters of Kuwahara filtering in case study 2.

Parameter	**Value**
Vertical window taper (%)	20
Vertical window half height (s)	0.004
Inline window radius (ft)	82.5
Crossline window radius (ft)	82.5

Table 4.E-17. Parameters of PNN supervised classification in case study 2.

Parameter	**Value**
Inline Increment	5
Crossline Increment	5
Vertical Increment (s)	0.02
Number of salt samples	5480
Number of not-salt samples	6368
Smoothing parameter r	1.0

CHAPTER 5: CONCLUSIONS

Although the three major applications of this dissertation are quite different, the unifying theme is the importance of data conditioning. Simply stated, the quality of the result from even the most mathematically elegant algorithm is only as good as the quality of the input data. In principle, least-squares migration provides the reflectivity in the subsurface, that when forward modeled, best reproduces the data measured on the surface. However, when the data area aliased, there are many subsurface models that can fit the given surface data. To address this issue, I've defined a least-squares migration workflow, controlled by a GUI, that provides a flexible way to add internal constraints to the conjugant gradient process. Such constraints can be as simple as a structure-oriented filter or as complex as a k_x-k_y filter to suppress acquisition footprint. Such filters allow the retention of long offsets in the input and steep dips in the output, providing improved images of fractured basement in Texas Panhandle survey.

Many filters are nonlinear, such that the order of their application is important. I found that in the presence of dipping faults, edge-preserving structure-oriented filtering sharpens the dipping fault edges. When analyzed on trace-by-trace vertical profiles, such sharpening introduces frequencies up to Nyquist that were not in the original seismic data, leading to impulse response artifacts in subsequent spectral balancing. In contrast, such artifacts are circumvented if I first spectrally balance the seismic data and then apply edge-preserving structure-oriented filtering.

Machine learning requires data conditioning as well. For the Canterbury Basin survey, I observed that all input attributes are non-negative and some attributes, such as coherent energy, are highly skewed. To analyze these data, the "preconditioning" involves scaling the data so that exhibit a simple mean and standard deviation. The simple linear z-score normalization works well for some attributes but does not account for kurtosis, skewness, or other deviations from a non-Gaussian distribution. I developed a data-driven nonlinear normalization that provides superior facies discrimination for simple projection and unsupervised classification algorithms. For supervised classification, I found better facies discrimination when the nonlinear normalization for each attribute also adapted to the distribution of each target class.