Interactive Seismic Imaging on a Multicomputer and Application to the Hosgri Fault

Thesis by

Charles Bruce Worden

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1992

(Submitted October 23, 1991)

Acknowledgements

I would like to start by thanking my advisor, Rob Clayton, for making this project possible. It was he who originally conceived the idea of an interactive imaging system and saw the possibility of implementing such a system on a multicomputer. He also arranged for the funds to make the idea a possibility. Rob has an uncanny ability to cut through to the crux of a problem and find a solution; his guidance often helped keep this project on track. I would also like to thank my other committee members: Dave Harkrider, whose humor helped keep things in perspective throughout my graduate career; Lee Silver, whose vast knowledge of California geology is amazing; and Chuck Seitz, one of the originators of the multicomputer.

I thank the individuals at Meiko Scientific, upon whose hardware our imaging system is implemented, for helping to make its acquisition possible. I especially want to thank Doug VanLeuven, whose persistence kept the deal alive despite a number of obstacles, and Robert Hardesty, who kept the machine running despite a different set of obstacles.

John Louie provided considerable help. While he was a student here, during my first year of graduate school, he helped get me started in seismic reflection. Later, while I was trying to finish my degree he provided useful discussions (by e-mail)

of the Hosgri images. John also provided his program "viewmat" for producing the PostScript plots that make up most of the figures in Chapter 3. Finally, John reviewed this thesis and made helpful comments and criticisms. I also appreciate John Vidale, another ex-fellow student, for supplying a version of his travel time code, saving me from having to write one from scratch when I was most pressed for time. Also, thanks to Neil Humphrey, who worked on an early version of the filesystem.

Doug Neuhauser, who was the systems manager here at the Seismo Lab, receives my deepest gratitude, for his friendship and his extensive help with this project. Doug participated in the process of selecting the hardware–even to the extent of flying to England with me for discussions with Meiko. Doug provided much help and many suggestions during the implementation of the system. He also took the Seismo Lab computer system from a state of near chaos and turned it into a reliable network, making the lab a more productive place.

I appreciate my "Chester House" roommates David Pickett, Mark Fahnestock, and Phil Ihinger, first for friendship and giving me somewhere to stay, and second for introducing me to the Lakers. The Lakers provided me with a much-needed diversion and got me out of the lab a few times a week. I could not mention the Lakers without acknowledging the greatness and heroism of Magic Johnson. Just two days prior to this writing Magic showed us that he is more than just a great basketball player; he put aside his pain and fears and doubts, and brought forward a message that needs to be heard, and he did it with his usual good humor and smile. True character shows in times of great stress.

I thank my classmates Ed Garnero and Brad Woods for helping to keep me sane as we struggled through our first two years of classes together. Some individuals might question the use of the word "sane" in the previous sentence, but fun we had and the laughs we shared made a difficult time survivable.

There have been too many other friends to mention here, and mentioning anyone runs the risk of forgetting others. However a number of people made my stay at Caltech more enjoyable: Doug Dreger, Tom Duffy, Rob and Jessica Graves, Scott King, Ann Mori, Richard Stead, Hong-Kie Thio, Dave and Lisa Wald, and Joanne Yoshimura, just to name a few.

Lastly, and most importantly, I thank Andrea Donnellan for her friendship, support, and love. I honestly don't know where I would be right now if it were not for her, but I do know that I wouldn't be as happy. Andrea also helped a great deal with the final preparation of this thesis, digging up references, making figures, proofreading, and encouraging me when I was ready to give up and move to Montana.

To *Andrea*

Abstract

Charles Bruce Worden, Ph.D.

California Institute of Technology 1992

Robert W. Clayton, advisor

Professor of Geophysics

California Institute of Technology

We present a system designed to change the manner in which seismic reflection data is imaged, by enabling interactive response to user input. This approach greatly eases the effort required to produce a seismic image and gives the analyst the flexibility to explore a wide range of models. We also argue that the ability to interact with the image can greatly aid in the interpretation process, and that the structural geologist charged with interpreting the image should be directly involved in the imaging process. Our approach differs from current seismic processing techniques that limit the ability of the seismic analyst to fully explore the imaging parameters. Current methods also provide the seismic interpreter with little information as to the robustness or reliability of the imaged structure.

The interactive imaging system is implemented on a heterogeneous, medium-grained multicomputer. This machine is configured to provide the substantial per-

formance required by the interactive imaging task. We discuss the implementation of the system as four separate, but interrelated tasks: data I/O, computation, image display, and user interface. Each of these functions is supported by hardware specifically suited to the task. The system software is designed to conceal as much of the parallel implementation as possible from a programmer wishing to add processing functions.

The interactive system is applied to a portion of EDGE seismic reflection profile RU-3 that crosses the Hosgri fault, offshore central California. From the imaged structure we infer that the Hosgri is a near-vertical fault, with relatively recent strike-slip displacement. We see no evidence, however, of recent thrust faulting.

Table of Contents

List of Figures

Alas, how swift the moments fly!
    How Flash the years along!
Scarce here, yet gone already by,
    The burden of a song.
See childhood, youth, and manhood pass,
    And age with furrowed brow;
Time was–Time shall be–drain the glass–
    But where in time is now?

—JOHN QUINCY ADAMS, *The Hour Glass*

Time goes you say? Ah no!
Alas, Time stays, *we* go.

—AUSTIN DOBSON, *The Paradox of Time*

Preface

Scientific jargon, while unquestionably useful for communication with other members of a particular field, is often completely without meaning to the uninitiated. In a document such as this, one can take one of three approaches to dealing with jargon: 1) one can use it freely with the assumption that no one but an expert in the field will ever read the document—an approach that almost guarantees that result, or 2) one can attempt to explain each term as it comes up—an approach that at best will bore, and at worst annoy the expert reader, or 3) one can use the terms freely, but provide a glossary that may or may not be of any use to the non-expert. As tempting as the first option is, we find that this document contains terminology from two distinct fields (multicomputers and seismic reflection), making it difficult to justify ignoring the audience. We therefore choose option three, and attempt to explain some of the more opaque terms contained herein in Appendix A.

For those readers interested in an introduction to multicomputers, we recommend an article by Seitz, *Multicomputers* [35]. Chapters 4 and 5 of *Introduction to Geophysical Prospecting* by Dobrin and Savit [7] provide a discussion of seismic data acquisition and reflection surveys. For readers wishing an introduction to seismic reflection processing we recommend *Seismic Data Processing* by Yilmaz [41]. Chap-

ter 1, section 4 provides a brief overview of the processing sequence.

C. Bruce Worden

November 1991

Chapter 1

Concept

> [The] discussion on migration leads us to a weak link between seismic data and geology. *That weak link is velocity.* We can be proud of the migrated stacked section as long as the vertical axis is time. However, when the geologist asks for a section in depth, we often hedge. Again, because of the uncertainty in velocity estimation, the depth section never is entirely reliable. The weak link of velocity is a fundamental problem in seismic exploration.
>
> —Özdoğan Yilmaz, *Seismic Data Processing*

1.1 Introduction

Seismic reflection data acquisition and processing is a billion-dollar industry (Goodfellow [12]), and that industry is directly dependent on computer technology. As acquisition systems grow larger and more sophisticated, they require ever more advanced digital computers to process the data. The development of methods of imaging the seismic data has also advanced in direct relation to advances in digital computing. Simply scanning the industry literature (The Society of Exploration Geophysicists' journal *Geophysics*, for instance) will show that it is a rare paper that

does not directly deal with the issues of computational efficiency or implementation of the technique under discussion. However, despite tremendous advances in computers, traditional techniques that have been in existence for over twenty years are still the principle means of processing the data. While the fundamental processes are sound, we assert that current computer technology allows an update in the way those processes are approached.

This thesis presents the conception, implementation, and application of an interactive system for the imaging of seismic reflection data. In the sense used here "interactive" means that the system output is generated quickly enough that it appears directly responsive to the user's input. The system, ISIS (Interactive Seismic Imaging System), provides the operator with the ability to selectively alter imaging parameters through a graphical user interface and to immediately see the results of those alterations in the form of an updated image on a video monitor. This procedure is in contrast to most current methods of imaging seismic data, which tend to involve static parameter selection and batch processing. Recently some systems have appeared boasting of "interactive" processing, but what is usually meant is that some level of parameter or process selection is interactive, followed by standard batch processing. ISIS is an attempt to jump away from that trend and provide truly interactive processing.

This chapter will discuss the motivation and the basic concepts of the ISIS project. Chapter 2 discusses the implementation issues, in general, and the implementation we pursued, in particular. Chapter 3 presents an application of the system to a reflection survey conducted off the California coast.

This project was partially motivated by the observation that the individual responsible for the interpretation of a seismic section—generally a structural geologist—rarely has any notion of what constraints can be placed on the image based on the processing that produced it. Similarly, the individual processing the data, the seismic analyst, often has little understanding of the geologic constraints of the region being imaged, nor an understanding of the features of the image that are of importance to the interpreter (e.g., the thickness of beds, the relationships between beds, the dip on structures). If the interpreter has direct control over the image, rather than working with a paper section, however, a much better understanding of the geologic structure can be developed. The interpreter, by adjusting the imaging parameters, can test the robustness of features, as well as the position, orientation, and relationships between reflectors. In addition, the geologist is able to apply geologic knowledge and intuition to the selection of imaging parameters—leading to a better-formed image. The ISIS project attempts to blend the roles of the geologist and the analyst.

1.2 Concepts of Interactive Imaging

ISIS provides a set of "tools" that facilitate interactive imaging while making use of well-known, robust imaging techniques. The three principle tools of the ISIS system are *movies*, *interactive focusing*, and *image deconstruction*. A fourth tool, interactive, graphical parameter selection requires little elaboration. It simply provides the user with an intuitive, user-friendly method of interacting with graphically displayed data by means of keyboard, mouse, or other input device.

Movies, as the name implies, are the rapid sequential display of multiple panels of seismic data sorted to some acquisition parameter (e.g., shot gathers). The theory behind movies is simple: the eye is capable of discerning small changes between two images as the images are flashed alternately in front of it. Astronomers used this technique for years in detecting nearby celestial objects against the background star field. The eye is also capable of picking out patterns that persist between two largely dissimilar or noisy images. Both of these abilities have proved to be valuable in seismic movies, the first for detecting lateral changes in structure and bad records, the second for discerning true reflectors from a noisy background. Movies allow the analyst the ability to inspect the entire data volume in minutes, and evaluate the effect of parameter choices on the data in the groupings that most naturally reflect the effects of those parameters. The ability to stop the movie, adjust parameters, reverse direction, or select another coordinate axis (e.g., receiver gathers) and continuing the movie are central to the concept. Note that animation-type speeds are not necessary for movies to be effective—rates of four to eight frames per second have proven to be adequate. Also note that unlike an earlier manifestation of movies in which the images were assembled, plotted, and photographed for later display through a projector, the movies discussed here are assembled "on the fly" and displayed on a video monitor.

The term "interactive focusing" is an analogy to a photographer looking through the viewfinder of a camera to focus an image. Some imaging parameters, the migration velocity, for example, have effects that manifest themselves only in a composite section, rather than the individual data gathers. Interactive focusing is a process by

which the analyst adjusts an imaging parameter, the image is recomputed, and the resulting image is displayed on the monitor. If the delay between the input and the update of the display is short enough to create a feedback loop between the machine and the analyst, true interactive focusing is achieved. To extend the analogy, current seismic processing techniques, are like pointing a camera in the general direction of the subject, taking a photograph, sending the film to be developed, and then making adjustments to the aim, focus, and exposure based on the photograph that comes back. ISIS allows our metaphorical photographer to look through the viewfinder of the imaging system.

The third concept, *image deconstruction*, allows the analyst to tie image points back to the data that produced them. For example, an analyst studying a feature on a stacked section can point to the feature and display the midpoint gather(s) that produced that portion of the image, and then access the shot or receiver gathers that produced the midpoint gathers. The analyst can further analyze the gathers using movies and interactive parameter selection. This technique provides the interpreter a method of distinguishing real structural features from processing artifacts, and for studying the geometric relationships that produce those features.

Each of the three concepts described in this section requires rapid and random access to the entire data volume, and a method of sorting the data into the appropriate gathers. The traditional approach using magnetic tapes to store the data is insufficient to provide the speed and flexibility required by ISIS. We devoted a great deal of attention to developing the appropriate functionality, as is discussed in Chapter 2.

1.3 Advantages of Interactive Imaging

The three basic imaging tools described in the previous section, coupled with the concept of geologist-as-analyst, constitute the basic principles of an interactive seismic imaging system. The advantages of such a system are numerous. As was already mentioned, having the geologist perform the imaging allows for the direct application of known geologic constraints to the processing sequence. The ability to interact with the image, parameters, and data allows for a better understanding of the image and, perhaps, a better interpretation of it.

A further benefit of interactive imaging is that the time required to process a typical seismic survey is reduced from the weeks or months required by traditional processing to just hours. This benefit derives from the same functionality that allows for interactive processing. In traditional processing, an analysis is made of the data and processing parameters are selected based on that analysis. The parameters are then applied to the data via a computer job running in batch mode, and typically a plot is generated showing the outcome. The parameters are then adjusted based on the previous iteration, and the job is resubmitted. The turnaround time for each iteration varies from a few minutes for simple processes, to several hours or days for complex ones. (The process is often slowed by other processes in the queue and by the need to mount tapes, often by the hundreds.) By definition, however, the turnaround time on an interactive system is often only a fraction of a second. Thus, each of the various processing steps can be accomplished very rapidly. The rapid turnaround has another advantage in that it affords a much greater use of the analyst's training and experience. In current practice the analyst must either idly wait for the next plot,

or switch to another project while waiting. Interactive imaging allows the analyst to concentrate on the task at hand and to pursue ideas as they occur.

The process of interactive imaging presents numerous opportunities to improve upon the images produced by traditional processing technology. The next several sections detail some of those improvements, which, when taken as a group, can lead to significantly better images.

1.3.1 Quality Control

Movies provide an excellent method of quality control throughout much of the imaging sequence because they allow the analyst to inspect the entire data volume, with various parameters applied, in a matter of a few seconds or minutes. Bad records stand out and may be edited from the data using a simple point-and-click interface. While it is difficult or impossible to inspect an entire data set using traditional techniques, it is a simple matter to do so with ISIS. Bad shots stand out in a movie of shot gathers; bad receivers show best in movies of receiver gathers (on land surveys) or constant offset gathers (on marine surveys). Random bad traces appear in any grouping.

Inspection of parameters is also easily achieved through the use of movies. Traditional processing often involves performing a particular analysis on gathers at a few evenly spaced locations along the line, and interpolation to the gathers in between. Interactive imaging carries this processing a step further by allowing the user to inspect the results on every gather in the survey. In locations where unaccounted-for lateral variations in the subsurface structure make the interpolated parameter incorrect, the user may stop the movie and correct the parameter. In this way, after

several passes through the data, the analyst has accounted for high frequency variations in the structure. For instance, surgical mutes may be applied to shot gathers, but if those mutes become ineffective due to lateral variations in the near-surface lithology, unwanted energy will creep into the image or excessive amounts of desirable energy are eliminated. Movies of shot gathers with and without the mutes significantly improve the quality of the parameters selected. Another example, this time with midpoint gathers, is the NMO velocity and the stretch mutes. The velocity analysis is perhaps the most important task of the seismic analyst. A movie of midpoint gathers with NMO applied allows the analyst to detect and analyze subtle variations in subsurface velocity. Through the use of movies, quality control becomes an integral part of many processing steps.

Another, planned, enhancement allows yet another level of quality control. When the geologist makes an interpretation of the seismic section, a velocity model is implicitly created as well. This model, can be fed back as the input model to the imaging process. If the model accurately reflects the subsurface structure, an image very similar to the original should be produced. This process can provide a direct check on the quality of an interpretation.

We anticipate that another level of quality control will be achieved if an interactive system is taken into the field where the data is being collected. Because the data can be processed quickly, field workers can partially process the data shortly after they are collected. The processed data may suggest altering the acquisition parameters, or the image may indicate structure which warrants closer study. Though difficult, it is much cheaper and easier to alter a survey while the equipment is in the field

than to send out another crew to resurvey the same site.

1.3.2 Fine Tuning through Interactive Focusing

Following completion of the initial processing steps, the technique of interactive focusing is applied to a composite image, i.e., a stacked or migrated section. The analyst can fine tune the velocity model to produce the sharpest image in areas of complex structure by working with the velocity model and observing the results in the finished image. Fine tuning allows for a level of analysis that cannot be adequately treated by conventional velocity analysis techniques. The focusing and defocusing of diffraction patterns in a migrated image is a prime example. An experienced analyst can easily identify a grossly over- or under-migrated image, but the ability to smoothly pass from one to the other allows for honing in on the proper velocity in much the same way a camera or telescope is focused by moving back and forth through the point of optimum focus in diminishing steps.

It may be argued that this technique gives the analyst the power to adjust the image to reflect some preconceived notion. While mistreatment is certainly possible, it must be remembered that the machine will honor the physics of the situation. If the analyst uses an unrealistic velocity to orient a reflector at some desired position or to create structure where there is none, other areas of the image will be adversely affected. Again, a level of quality control is available where none existed before. If slight variations in the velocity cause the image to be radically changed or structures to vanish, the interpreter must regard those features with suspicion. On the other hand, robust features may be distorted by such tests, but will retain much of their character even through relatively major changes in the model.

In the prototype system, interactive focusing, while fully implemented, is only truly interactive on relatively small data sets. Effectiveness on larger data volumes will require more computing power and, perhaps, hand-coding of the most CPU-intensive routines. Ultimately, even fully interactive pre-stack migration may be available.

1.3.3 Exploration of the Parameter Space

The ability to interactively select parameters and observe the effects of those parameters on the data allows the analyst to fully explore the parameter space. These parameters (mutes, filter settings, velocity models, statics, etc.) have a direct bearing on the image produced. Simple fine tuning of parameters may find locally optimal solutions, but globally optimal solutions may require more drastic alterations. Interactive imaging gives the analyst the tools to explore various possibilities as they present themselves and to return to earlier models if the results are unsatisfactory. Another advantage of ISIS is the ability to explore the relationships between parameters, and the trade-offs they represent. The ability to thoroughly explore the parameter space is another instance of making better use of the analyst's extensive knowledge and experience. And, again, the result may be an improved image or interpretation.

We applied ISIS to the study of the Hosgri fault zone (Chapter 3) and were able to explore various velocity models ranging from models with smooth velocity profiles to ones with steep velocity gradients near the fault zone. Exploring a broad subset of the parameter space proved to be important to understanding the structure of the fault.

1.4 Other Features of Interactive Imaging

Because the processing is driven by input from the user interface, it is possible to record the user's input and replay it at some later time. In this way a self-documenting processing sequence is created. Another user can replay the sequence to verify the analysis or determine the current status of a project. The parameters are stored in files at various stages of processing, so the new user may attempt a different avenue of approach to the processing without corrupting the previous work. Thus, even a finished project may be easily re-analyzed and new ideas may be tested. This functionality also provides another means for quality control in an industry setting where a manager may wish to check an analyst's work.

Interactive imaging may also serve as an educational tool. Processing techniques and the effects of parameters may be explored easily and in a way that demonstrates the important issues.

As discussed in Chapter 2, we designed ISIS to allow for the easy addition of new processing functions. Thus, the system is ideally suited for those wishing to test new algorithms or imaging techniques. The new function can be easily integrated into the existing system through the standard interfaces provided (also discussed in Chapter 2) and tested on a variety of data and with various choices of parameters.

Through development of ISIS some unanticipated advantages of the system became apparent. For example, when we studied midpoint gathers with NMO applied, we knew that such movies would easily locate areas of unaccounted-for lateral variations in velocity, but we did not expect them to aid in distinguishing diffractions from primary reflections in areas of complex structure. The diffractions, however, can be

seen moving up and down in the gathers and through other layers, making them easy to distinguish from the primaries. This distinction enabled us to concentrate the velocity analysis on the true reflectors.

1.5 Why Interactive Imaging?

Despite the forgoing, some individuals may question why we bother with interactive imaging at all. Why not just automatically process the data and be done with it? The answer is twofold. First, it would be difficult to conceive of a fully automated system to process seismic data. There is no generally recognized method of computationally determining the accuracy of a seismic image, or for preferring one image over another. Lacking such a method, it is impossible to automate the task of adjusting the parameters to optimize the image. The enormous complexity of geologic structures and the recorded data make such a method an unlikely near-term development. In a similar manner it is difficult to imagine a generalized inversion formula for seismic reflection data, since the tradeoff between the reflectivity and the velocity structures of the subsurface is generally not completely constrained by the data. Now and for the foreseeable future, the expertise and judgement of a human analyst will be required for the accurate imaging of seismic data. By exploiting greater computer power, however, we can facilitate a wider exploration of the parameter space, which provides us with greater control over the non-uniqueness of our models and enables us to constrain the possibilities presented by the data. The second reason to not automatically process the data is that even if an automatic system were available, it would not provide the geologist with the advantages described in

the previous sections. The interpreter would remain in the position of making the structural interpretation with only a static, paper section, without an understanding of the processes, parameters, and data that produced it.

Chapter 2

Implementation

2.1 Introduction

In this chapter we discuss the implementation of the Interactive Seismic Imaging System (ISIS) described in the previous chapter. We first discuss our general approach to the problem, the performance characteristics necessary to provide the functionality described in the previous chapter, and the specific hardware platform on which we implemented ISIS. We then discuss the software design issues and software implementation. Finally, we describe the current state of the seismic processing software, and proposed future additions.

2.2 General Design Considerations

Seismic processing functions consist of four principle tasks: parameter input, data input, computation, and data output. The seismic analyst is responsible for the selection of the processing parameters appropriate for the data. The data input consists of some portion of the seismic traces recorded by a field survey. The com-

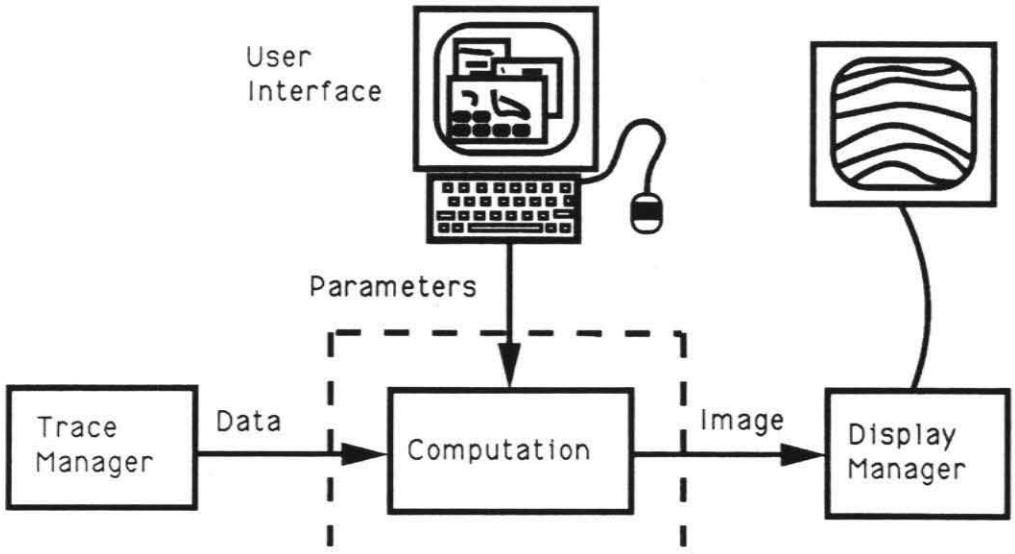

Figure 2.1: Imaging tasks. The four principle divisions of the ISIS system are shown. The dashed lines represent software layers that insulate the computational processes from the other functions.

putation function(s) processes the data as specified by the parameters, and produces the output, usually an image of some form. Each of the four parts has its own particular set of requirements and design issues. Keeping this fact in mind, we divided the imaging task into four components (Figure 2.1): 1) a computational engine to produce the seismic images, 2) a seismic trace manager to provide the data to the computational engine, 3) a display manager to receive images from the computational engine and display them on a video monitor, and 4) a user interface to provide parameter selection and control functions. The function and implementation of each of these components will be discussed in detail in subsequent sections.

For reasons discussed later, ISIS is implemented on a multicomputer (or parallel

computer). Implementation on a multicomputer has a number of consequences that we had to consider during the design process, not the least of which is the added complexity of programming this type of machine. To simplify the implementation process, the ISIS software is divided into two principle layers: the system layer, and the application layer. The system level software provides functional interfaces to the four main components, and conceals implementation details from the application. The application software consists of the seismic imaging functions (i.e., those that process seismic data), and the user interface. The ISIS system presented here consists of both—it is a processing system built upon the system software. There are several reasons we made this division. First, the system had to be programmable, so that imaging or user interface functions could be added quickly and easily, without requiring changes to existing functions. Second, it was desirable that the system be as portable as possible. This goal is difficult to achieve considering the nature of the hardware, but using system software to hide implementation details simplifies porting the user interface and the processing functions. This approach also isolates implementation-specific software in a few locations, thus easing the porting effort of the system software—a significant advantage considering the rapidly changing and non-standardized nature of parallel hardware and software. Third, the application programming effort is greatly eased by the concealment of the implementation details, and while the parallelism of the computational processes cannot be entirely concealed from the applications programmer, the parallelism of the trace manager and the display manager can. In this way the applications programmer is free to choose the algorithm that best suits the task without having to consider the imple-

mentation details of the system.

2.3 Performance Considerations

Each of the principle components of the system must be capable of providing a certain level of performance in order to achieve the functionality described in Chapter 1. For example, in order to display movies at eight frames per second the graphics unit must be able to absorb and display approximately eight megabytes of data per second, assuming one megabyte per image.

Other performance requirements are not as easy to quantify. If we consider a moderate sized survey of 250,000 traces each of 1000 time points, the trace manager must be able to store one gigabyte of data and retrieve traces randomly distributed throughout the data volume. The most demanding application for the trace manager is interactive stacking—the entire data volume must be delivered, in midpoint order, in approximately one second, to maintain minimal interactive capability. Assuming that the data are stored on disk, we require a sustained 1 Gb/sec transfer rate. More imposing, however, is the requirement of 250,000 individual disk seeks in the same time period. At 2 Mb/sec a common disk drive takes 2 ms to transfer a 4 Kb record, but the average seek and rotational latency times add up to about 20 ms. Thus, about 5000 drive-seconds are required to transfer the data. This requirement is overstated, however, in that the interactive stacking does not have to restack the entire image, but only that portion affected by a parameter change, perhaps ten percent. In addition, the data, if spread among several disks will not span the entire media, and thus the seek time will be closer to the minimum than the average.

By adding an intelligent caching scheme as well, the performance requirement is on the order of 50 drive-seconds/second. Note that if the entire data volume, or a large fraction of it, can be stored in computer memory, the performance of the trace manager can be greatly increased.

The expense of interactive post-stack migration, again with moderate one-second updates, is the most demanding of the computational tasks. Consider the survey described above. The stacked section would have approximately 2000 midpoints each of 1000 time points, i.e., about two million image points. If we assume 500 floating point operations per image point, we require a sustained performance of at least one gigaflop to achieve the interactive migration. Greater performance would be required by more sophisticated migration algorithms.

Interactive prestack migration would be considerably more computationally intensive, and would require the trace manager to supply the entire data volume at least once per update cycle. The computational and data requirements of interactive prestack migration are so high, however, that we must relegate it to the "future developments" category. Teraflops performance is required, and while machines with this level of performance will become available over the next few years, it will be some years more before they are within a price range that will make them available for this type of work.

The performance requirements described above, while substantial and difficult to obtain in current sequential machines, are readily available in current multicomputers. The I/O requirement of 50 drive-seconds/second can be achieved by fifty disks working in parallel. The Gflop computational performance can be achieved by

50 separate 20 Mflop processors working together. The graphics requirement can be achieved by chaining together several graphics units. Thus, implementation on a heterogeneous multicomputer was a logical choice for our system. Other options do exist, but they generally fall short in one or more respects. A state-of-the-art supercomputer with an advanced I/O subsystem might be able to achieve similar performance numbers, but the cost would be beyond most budgets, and the system would not be scalable, i.e., there are situations in which it would be desirable to have both a larger system (in a commercial processing center, for example), or a smaller system (for field use). Another option, a custom-built hardware system, would suffer from both of these problems, as well as the difficulties in maintenance and in reproducing the system at other sites.

2.4 Physical Implementation

We carried out an extensive search to find a hardware system capable of providing the performance and functionality necessary to implement the ISIS prototype. The principle requirements were that the machine provide graphics output at approximately eight frames per second, that multiple, scalable disk I/O was available, that significant, and scalable computational speed could be achieved, and that some controlling process could drive the system. The search quickly focused on a few parallel machines, both of the message-passing and shared memory variety. We ultimately selected a machine manufactured by Meiko Scientific Ltd., of Bristol, England. Figure 2.2 is a schematic diagram of the system hardware. The trace manager is implemented on eight nodes, each of which consists of an Inmos T800 transputer as the

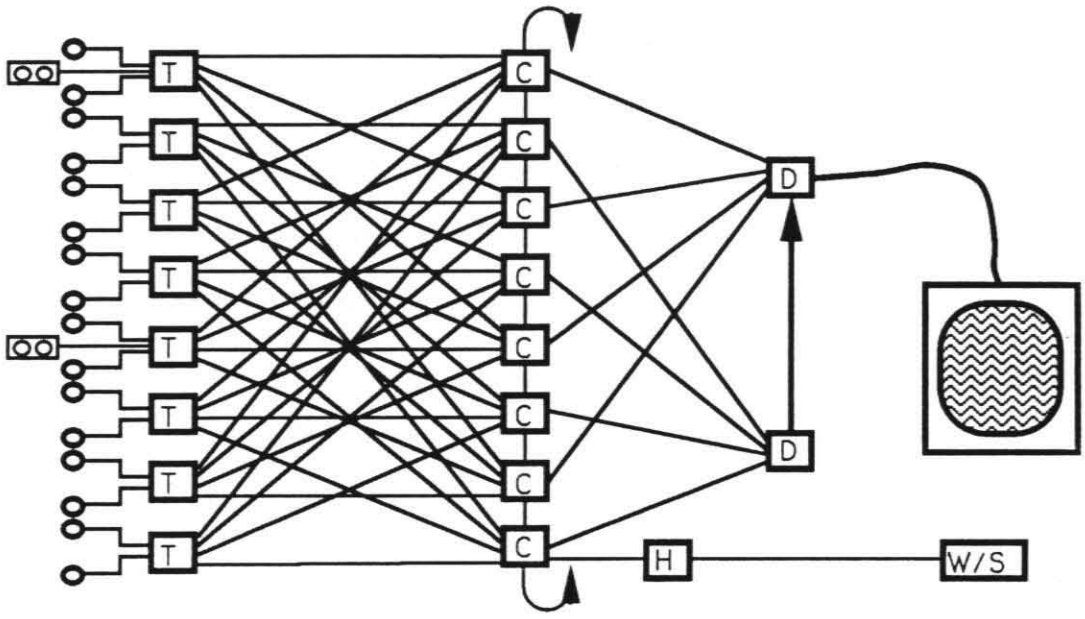

Figure 2.2: Layout of processes on the Meiko multicomputer. Each box enclosing a letter represents a node: trace manager processes are marked "T", computational processes "C", and display manager processes "D." "H" is the system host board, and "W/S" represents the Sun workstation, where the user interface resides. Each trace manager node has access to two disk drives (small circles), and two nodes also have 8mm tape drives. The lines between nodes represent communications channels.

principal processor. Though the T800 is a relatively slow processor by current standards, the performance requirements of the trace manager process are not as severe as those of the computational processes, and the T800 has proven adequate. Each of the trace manager nodes has 8 Mb of main memory and is linked to a SCSI bus. Two one-Gb disk drives are attached to each bus (16 total), and two of the processors have 8 mm tape drives that provide an efficient means of loading data into the system. The T800 is configured with four bi-directional communications links over which it can send messages to other processors. The T800's communication bandwidth is unimpressive (~1 Mb/sec per link, each direction), but the message start-up latency is very low (under 1 ms). The Meiko message-passing software, CSTools, effectively insulates the applications programmer from needing to know how the communications links are interconnected, providing point-to-point communications regardless of the actual path the message takes. The software also provides for a large number of logical communications links, but since there is a significant performance penalty for each "hop" a message must make between source and destination, there is some advantage to configuring the links to minimize the distance between communicating processes. Meiko provides software to accomplish this task at program boot time.

Another group of eight nodes provides the computational ability. Each of these nodes consists of a 40 MHz Intel i860 processor and two T800 transputers. The i860 provides the computational power and the T800's provide communications services. This separation is invisible to the applications programmer—the same code will run transparently on the T800- and i860-based nodes; the system software deals with the details of the configuration. The result is that the i860 "looks" like any other

processor in the system, except that it has eight hardware communications links. Each of these nodes is configured with 8 Mb of memory. In addition, there are 2 Mb of memory shared between the T800's and the i860 for communications support.

Two more nodes provide graphics support. Each of these nodes is configured with a T800 processor, 4 Mb of standard memory, and 2 Mb of video RAM. In addition to the transputer links, the nodes are connected by a "pixel highway" that links the video output of the two boards. The master board is configured with an RGB analog output that drives a color monitor. These boards can operate in two modes: 1) each board in the chain provides one segment in the image (e.g., in the case of two boards, one board would be responsible for the upper half of the image, the other would be responsible for the lower half), or 2) the boards may each produce an entire image, and the application chooses the board to be displayed at any given time. The boards may also flip between the first Mb of video memory and the second Mb, providing a double-buffering mechanism. The limited computational performance of the T800 is not a severe handicap in this application, since there is little computation to be done. In fact, the T800 has a machine instruction for copying one memory region to another, making the process of moving an image from program memory to video memory extremely efficient. In most other senses, however, the nodes are fairly unsophisticated—there is no hardware vector drawing, for instance, nor any other specialized graphics rendering functions. Nevertheless, the node has proven to be adequate for the purposes of this application. More of a handicap is the limited 4 Mb of program memory; 8 Mb would be much less confining, but was unavailable.

Additional hardware connects the Meiko system to a Sun workstation. From the

point of view of the applications program, the Sun appears as another node in the system. The Sun also supplies system services, and a development environment. The user interface is implemented on the Sun, giving the applications developer a number of software options from which to choose. The communications between the Sun and the other nodes is less efficient than the node-to-node communication within the Meiko system (about 200 Kb/sec), and there is only a single link, thus large data transfers are inefficient and should be avoided. The Sun also has the opposite sense of byte ordering than do the nodes in the Meiko, so special attention is needed to assure proper communications. We address this topic in Section 2.5.3.

As mentioned earlier, there is a performance penalty for each additional hop a message must make from source to destination, thus to achieve maximum performance, this distance must be minimized. In addition, since there is limited bandwidth on the links, message traffic needs to be distributed as evenly as possible to avoid bottlenecks. As discussed in Section 2.5.1, each computational process, in general, receives data from each of the trace manager processes, thus complete interconnectedness between these two types of processes is required. Figure 2.2 shows the ISIS configuration. Four of the eight links from each of the even-numbered computational nodes are used to connect to the even-numbered trace manager nodes and, similarly, the odd-numbered compute nodes connect to the odd-numbered trace manager nodes. Two of the links from each computational node are connected to its nearest neighbors, forming a ring of computational nodes. In this way data from any trace manager node can reach any computational node in a maximum of two hops, and the traffic is distributed evenly. One more link on each computational node is

used to connect to a graphics node, again in an even-odd configuration. There is still one free link on each of the i860 nodes, which may be used for larger hardware configurations. For example, the current number of nodes could be doubled, the second half wired exactly as the first, and the two halves connected by the extra links. In this way the power of the machine would be doubled at the cost of only a single extra hop in the maximum trace manager-to-computational node distance.

An advantage of the Meiko machine over some other machines is that it is expandable by any incremental number of nodes; it is not restricted to a power of two like many of the binary n-cube machines. Thus, if some unacceptable imbalance is found between the components, more nodes of a certain kind may be added to even the load distribution. Thus, our software design must take this possibility, as well as the issue of general scalability, into account.

2.5 Software Design

The development of a seismic imaging system from the ground up would seem to be an almost overwhelming task. Commercial systems often employ dozens of programmers and require years to complete. The ISIS project, with its emphasis on interactive processes and parallel implementation, would seem even more forbidding. And yet we were able to implement the prototype system in a little over a year with only one person consistently working on the project. We accomplished the task by eliminating some of the major software engineering obstacles that confront the implementors of seismic imaging systems.

ISIS has no facilities for handling tapes (except that which is necessary to load the

data from 8 mm tapes onto the system's disks). The user is expected to provide the seismic data in the proper format, with the proper headers, on the tapes. Handling of tapes of differing formats, running on differing devices and platforms requires a substantial software effort, one which we avoided. There are also no facilities for generating hard copy plots; an image displayed on the monitor may be dumped to a file, and the user may then plot it by whatever means is available. Again, providing a portable plotting package for a large variety of devices is a complex, expensive task.

Commercial seismic processing systems also supply a considerable number of auxiliary functions for performing various operations on the data. Many of these functions are rarely used, but are required to round out a commercial system (e.g., convolution, correlation, derivative operators, etc.). Others are useful, but not required for basic processing (deconvolution, F-K filtering, etc). We did not attempt to implement such functions, choosing instead to concentrate on the core processing functions.

We also simplified the programming effort through the modularity of the code. By isolating the various functions, as described later, we avoided some of the well-known non-linear growth in programming complexity as the system grew. In addition, we took advantage of standard software packages wherever they presented themselves. We used Unix utilities, X-Windows libraries, Meiko's CSTools message-passing libraries, Sun's XDR package, Sun's XView library, etc. Some of these choices hamper portability, a serious issue, but greatly eased the development effort.

As another simplifying design decision we avoided any attempt to break new ground in developing seismic processing software. In most cases, we used well-known,

well-tested algorithms and implemented them in a straightforward manner. In some cases this approach did not produce the most efficient code, but for our purposes the speed of implementation offset that loss. The simplicity of the interface to the trace manager (Section 2.5.1) also greatly eased the implementation of the processing software. For example, the requesting process can specify that the trace manager deliver traces with statics and NMO applied, therefore a stacking function might consist of only a few lines of code that request a series of midpoint gathers and sum each trace into the stack.

Finally, while Fortran is the dominant language of the seismic processing industry, we chose to implement ISIS in C. The reason for this choice is threefold. First, in a message-passing system as complex as ISIS, data structures are regularly packed into byte streams, sent as messages, and unpacked into data structures by the receiving process, often with a level of byte swapping in between. C is clearly superior to Fortran in manipulating complex data structures. Second, most windowing software (e.g., X, SunView) are written for C. Using C for the user interface and Fortran for the computation would have added more complexity to an already complex system. Finally, we have observed that much of the software for parallel programming is written in C first, and Fortran interfaces are added later. This fact reflects the basic appropriateness of C, or rather, the inappropriateness of Fortran, for the purpose of message passing. Since we wanted to take full advantage of the available libraries, C was preferable.

Once we decided to pursue parallel implementation of the application, two interrelated sets of issues became relevant. First were the general issues of parallelizing

the application, and second were the issues of scalability. Most applications can be divided in more than one way; it is important to select a way that suits the underlying hardware, but that is flexible enough to change with hardware changes. Even within the class of medium-grained message-passing multicomputers there is considerable range in the message-bandwidth-to-compute ratios, and in the message start-up latency. Closely tied to this issue is the issue of scalability. The ideal implementation of an algorithm would scale smoothly from one to many thousands of processors and require little communication between each instance of the process. Also, the performance of such an application would ideally scale linearly with the number of processors. Algorithms of this sort are rare, and often some concessions must be made to enhance performance on the available hardware at the expense of generality.

Seismic data have particular levels of granularity that the applications programmer would be unwise to ignore. A typical seismic survey consists of a collection of several hundred source gathers (which can be reordered into a small number of other gather types). Each source gather is made up of about 100–1000 seismic traces. Each trace consists of about 1000 time samples recorded by a geophone. Common seismic processes have developed with natural input-output relationships that take into account this style of data storage. Most algorithms are developed to deal with the data by gather or trace. Stacking, for instance, generally works on each CMP gather sequentially, applying NMO to each trace and summing the traces together. There is no fundamental reason that the process cannot work on other types of gathers (as long as each trace is summed to the correct midpoint trace), or even on the

individual data points, but for efficiency and simplicity of implementation the CMP gather approach usually works best. Similarly, the output of most seismic processes consists of a record section made up of traces of time or depth data values. This output is usually displayed so that the horizontal axis represents surface position and the traces run down the vertical axis with time or depth increasing downward. Output functions may be efficiently designed by keeping this format in mind.

Because of the nature of the recorded data and the way it is processed, seismic imaging lends itself well to medium-grained multicomputers. Machines of this type generally have no more than a few hundred, relatively powerful, processors, each of which has a few megabytes of memory. A machine with many times this number of processors would require a general rethinking of the processing model. For example, to perform a CMP stack with fewer output traces than available processors, each process would have to work with a fraction of an output trace while still requiring access to the entire input gather. There are inherent inefficiencies in forcing the seismic data to be more fine-grained than its natural state. We have therefore restricted our efforts to algorithms that scale well from just a few processors—preferably a single processor—to a few thousand processors at the most.

2.5.1 Trace Manager

The principle purpose of the trace manager is to provide the application with a means of requesting and receiving seismic traces in an organized manner. Our main goal in designing the interface to the trace manager was to conceal implementation and implementation-specific details from the application by providing a simple, uniform interface across all implementations. Each instance of the trace manager process

must be fair to all of the requesting processes (i.e., no requesting process or group of processes may lock out other processes from receiving requested data). Another requirement is that any request for data must be guaranteed to finish in a finite time (assuming that the receiving process is accepting traces). We will discuss the interface and implementation of the trace manager in the next two sections.

Applications Interface

Before any process requests data, the trace manager processes must be initialized. It is the responsibility of the user interface process to do the initialization by calling the function *datainit()*. The arguments to *datainit()* include the names of the file containing the data and another file containing auxiliary information on the stored data. If no combined master list of this auxiliary information exists in the user interface's current directory, one is assembled and placed there (by collecting and merging a copy of the auxiliary lists of all of the trace manager processes). This master file is read by *datainit()* and the information is available to the user interface for mapping, selecting data, etc. A related function, *enddata()*, releases the data structures and leaves the system in a state in which another call to *datainit()* may be made, perhaps to work on another data set.

The principle interface to the trace manager consists of two functions: *datarequest()* and *getdata()* (a third function, *getrequest()*, builds a default request structure). The data are available in one of a small number of common groupings (i.e., shot, receiver, midpoint, and constant offset gathers), by designated individual trace, or by the entire data volume. The application fills in the request structure returned by *getrequest()* with the appropriate geometric information and then calls *datare-*

quest() with the request structure as the only argument. The application then loops over calls to *getdata()*, each of which returns a seismic trace and an associated header until no more data of the requested type exist, at which point calls to *getdata()* return the value zero. It is an error for a process to call *datarequest()* while there are still traces outstanding from a previous call. The traces returned by *getdata()* may arrive in any order; the application must be designed with this fact in mind. To facilitate this approach, the header that arrives with each trace contains trace-specific information: the x, y, and z coordinates of the source and receiver, midpoint coordinates, source and receiver flag numbers, the receiver sample rate, etc. A sample function that makes use of this interface may be found in Appendix B.

These functions present a simple interface for the applications programmer that requires no knowledge of the number of trace manager processes. The interface would be the same whether it was implemented on a single uni-processor machine, a network of workstations, a shared-memory multiprocessor, or a message-passing multicomputer such as the Meiko. In this way a level of parallelism is completely hidden from the application, greatly simplifying the programming effort.

Closely associated with the basic trace manager functions is the concept of *lines*. Lines are defined by the user in terms of the map coordinates of the survey. Each line has associated with it an off-line tolerance, and binning interval. The off-line tolerance specifies a distance perpendicular to the line, on either side of the line, within which a source, receiver, or midpoint is considered to fall on the line. The binning interval, combined with the off-line tolerance, specifies a set of bins used for midpoint sorting. Lines may contain multiple segments (to deal with irregular

recording geometries). All requests for data must have associated with them a predefined line. Any request for data will only apply to the data that falls within the boundaries of the line designated in the request. A small number of functions are provided for defining and accessing the lines.

The lines mechanism provides the user a way of defining reconstruction lines through surveys with complicated recording geometries, as well as providing a means of focusing the scope of an analysis. Multiple lines may be associated with a survey, and the user is free to choose from among them at will. Many parameter types are associated with a particular line. For example, the velocity models each apply only to a specific line, though any given line may have multiple velocity models associated with it (only one may be active at any given time, of course).

System Level Implementation

Each instance of the trace manager process (Figure 2.3) consists of a data archive upon which a subset of all the traces in the survey are stored. The traces are divided as evenly as possible among the trace manager processes, but need not be stored in any particular order. Each instance of the trace manager has a unique subset of the traces which does not intersect with any other instance's subset. In this implementation, the data are stored on magnetic disk, but other storage means are possible. Each process maintains a list of the traces stored in its local archive. This list includes the location of the trace in the archive, as well as indices into source and receiver coordinate lists. These lists are generated by a utility program that is run only once, after the data is loaded onto disk. The lists and a small amount of auxiliary information are written onto disk, and are read to initialize the trace

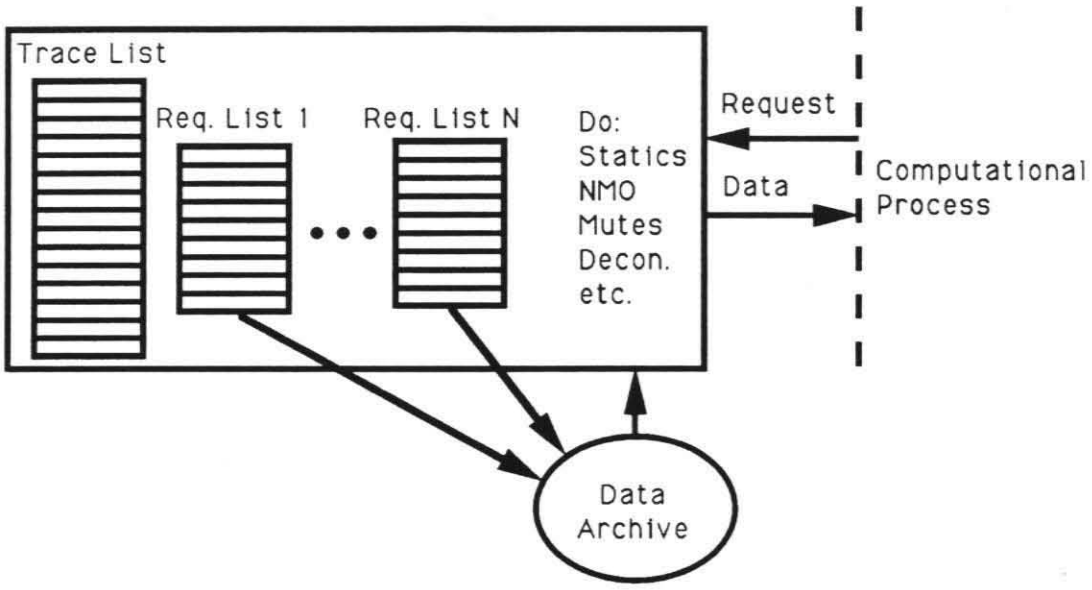

Figure 2.3: A schematic representation of an instance of the trace manager process.

manager when *datainit()* is called.

When a request arrives from a computational process, each instance of the trace manager searches its list of traces, indexing into the source and receiver lists for the appropriate geometric information. A secondary list of traces that satisfy the request is generated. As mentioned above, there may be more than one computational process requesting data at any time, so there is a need to assure that each computational process is treated fairly, and that all requests are guaranteed to finish. To accomplish these requirements, the trace manager process selects the first item from each of the lists and designates a buffer for an asynchronous read of the required trace. Multiple buffers may be queued from each list as memory permits. When a read is complete, the header is prepared and attached to the beginning of the buffer

containing the trace, and the buffer is queued for asynchronous transmission to the requesting process. When the transmission is complete, the trace manager selects the next item from the list corresponding to the trace just transmitted and queues it to be read into the free buffer. When no traces from a given request remain to be sent, the trace manager sends a token to the requesting process indicating the end-of-data condition.

When the application calls *datarequest()*, the ISIS system software in the requesting process immediately sends a copy of the request to each of the trace manager processes, and queues buffers to receive incoming data. As each call to *getdata()* is made, the function waits for the first available incoming trace. When a trace arrives, the header and data are copied to user-supplied addresses, and *getdata()* returns a positive integer. As end-of-data indicators arrive from the trace manager processes, the appropriate buffers are dequeued. When all of the buffers have been dequeued, the request is exhausted, and *getdata()* returns zero.

The application may request that certain operations be performed on the data prior to its delivery by *getdata()*. Simple, well-known operations such as statics, mutes, normal moveout, and deconvolution may be applied as long as system functions exist to perform them and the necessary parameters have been defined. The application sets flags to request that one or more of these operations be performed through the function *setdataops()*. Note that the operations, while handled by the trace manager system-level software, may be performed on any processor. Though the most common location to perform the tasks would be the sending or receiving process, in some implementations an intermediate processor might be used. In this

implementation, these functions are performed by the receiving process on the computational node. This choice was made because of the i860's superior computational performance.

The performance of the trace manager could be enhanced by ordering and combining read requests, caching data, etc. No attempt was made to do so in the prototype system, primarily because of time constraints. Future work in this area could lead to significant improvements in system performance.

The I/O software that Meiko provided for the SCSI nodes allows for synchronous or asynchronous reading and writing of the disks as raw devices, but did not include a filesystem. We implemented an extremely simple filesystem that provided coarse-grained disk striping across all the disks on a bus. The disks combine to form one large logical device and the striping is implemented by defining a logical block size for each file. When writing, the first available block is written until full, and then writing switches to the next disk on the bus, eventually cycling back to the first. The purpose of this implementation was to allow for the selection of a logical block size equal to the size of the data traces (rounded up to the nearest multiple of the physical block size of the device), so that no trace is split across disks. In this way, the data appear to be in a single file (simplifying the development task), but the various reads take advantage of overlapping seeks. Since the size of the logical device may exceed the limit of a 32-bit unsigned integer, the offset for reading and writing is specified in logical blocks. Files span continuous blocks and are written end to end. To remove a file, all of the succeeding files are moved down to fill the space. We implemented a number of utility functions to deal with the filesystem, as well as

functions for loading the data from tape to the disks.

2.5.2 Display Manager

The purpose of the display manager is to provide a means for the application to produce graphical output. As with the trace manager, one of the principle design goals was to conceal implementation details from the application. As the system has developed, the display manager has undergone several major changes. As of this writing the interface has stabilized, and the program will probably undergo a rewrite in the near future to clean up the interface so that it conforms more strictly to the description below.

The display manager handles a task that is somewhat more complex than that of the trace manager. First, the user interface requires direct knowledge of what image is currently being displayed. In order to maintain synchronization with the images, the user interface must authorize the display of any image. Second, the display manager, upon receiving a trace for plotting cannot know *a priori* in what position to plot the trace. With no outside help it would have to wait until all traces had arrived before deciding on a horizontal scaling factor. This requirement for *a posteriori* knowledge is inefficient, since the trace manager must wait idly for all of the traces when it could otherwise be preparing the image, and cumbersome, since it requires the display manager to buffer and sort traces. Finally, in the case of the trace manager the $N \times M$ interconnectedness (where N and M are the number of trace manager and computational processes) reduced to N identical, independent trace manager processes and M identical, independent computational processes. In the case of the display manager, however, P display processes are required to perform

a particular task for M computational processes, but the general solution requires coordination among the display processes, and each must coordinate with $\frac{M}{P}$ computational processes.

Applications Interface

In an attempt to mirror the interface to the trace manager, we designed the display manager to present the application with two principle functions: *draw_trace()* and *plot_data()*. The application repeatedly calls *draw_trace()*, once for each trace to be plotted. The arguments to *draw_trace()* include the trace to be plotted, the header, and a flag indicating which of a small number of plot types is to be generated. When the plot is complete, the user calls *plot_data()* with an argument indicating whether the image is a complete image unto itself, or whether it is one segment of a composite being produced by the combined efforts of all the processors. Another argument allows the application to erase the entire existing image before plotting the new one, or to simply overwrite the existing image in the places where the two overlap. When *plot_data()* returns, the application is free to start producing another image. An example function making use of these functions can be found in Appendix B.

All scaling is handled within the display manager. The application has control over the various scaling options from the user interface, where a function is provided for modifying the controlling structure. Various parameters include the horizontal scaling for the various types of gathers, the vertical scaling, the positioning of the image, the method of scaling the data points, automatic gain control, and so on.

The user interface also controls the actual display of images. The *plot_data()* function signals the display manager that the image is complete, but the user interface

is still required to send a token to the display manager to signal that it should display a particular image. The token consists of information indicating the source of the image and the type of image. The user interface application is also provided with a function to determine whether the display manager is ready for the next token. This function helps to prevent system lock-up in the event of an error or slowdown somewhere in the system. While this system is a bit awkward and requires careful bookkeeping of tokens and images, it does allow the user interface a deterministic knowledge of the current state of the display.

While a movie is being displayed, the application is looping over multiple calls to *draw_trace()* and *plot_data()*. In this situation, the *plot_data()* function will not return control to the application until subsequent calls to *draw_trace()* can be accepted (there is limited buffer space—see next section). In this way the system will be gracefully slowed when the production of images exceeds the rate of display. The user can kill a movie-in-progress by sending a special token. This token kills messages in transit, clears image buffers, and flags the application with a special return value from *plot_data()*. Upon sensing this return value, the movie application is expected to terminate and return to the calling function.

System Level Implementation

When a particular data set is initialized for the first time, the user interface process inspects the trace information returned by *datainit()* (see Section 2.5.1) and makes an initial estimate of the appropriate scaling factors for the different plot types. This task is done with the expectation that the user will adjust the scaling as needed. The system software saves the scale settings from one session to the next. By having the

initial scaling factors provided, the display manager can process the traces as they arrive.

Each trace that arrives in the display manager via the *draw_trace()* function is scaled as specified by the user interface, and written to one of two image buffers maintained by the manager. (Other implementations may have more than two buffers, or only one.) When a call to *plot_data()* is made, the image is marked for display and subsequent calls to *draw_trace()* will write to the other buffer. Another call to *plot_data()* will not return until the first image has been displayed, transmitted, or otherwise discarded.

As with the trace manager, we allowed the location of the processing functions to be flexible to take into account the needs of different implementations. In this implementation, the image buffers reside on the computational nodes, and the trace scaling and plotting is done there. There are several reasons we made this choice. First, as with the trace manager processing functions, the i860 nodes are substantially faster than the T800-based graphics nodes. Second, there are more computational nodes than graphics nodes, thus the computational load is distributed not only to faster processors, but to more of them. Thirdly, since the image is an array of single bytes, but the data is stored as four-byte floats, the transmission between the computational nodes and the graphics nodes is reduced by a factor of four. Finally, space is limited on all nodes, but especially so on the graphics nodes, where only 4 Mb of program memory is available. As it is, one of the incoming image buffers must be stored in the inactive megabyte of video memory. If implemented on the graphics nodes, the double buffering scheme described above would require

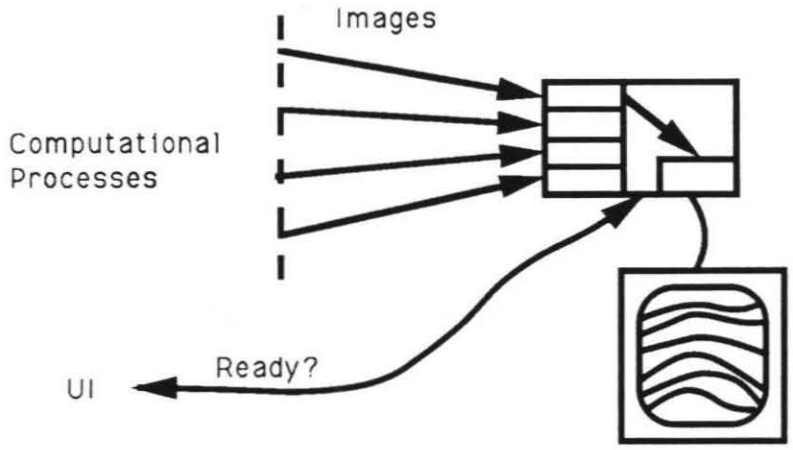

Figure 2.4: A schematic representation of an instance of the display manager process.

a minimum of 8 Mb of memory (assuming a system with two graphics nodes and eight computational nodes), plus space for the program and system software. On the i860 nodes, however, the buffers only require 2 Mb of the 8 Mb available. In the event that a computational process requires more memory than is available when it is called, functions are provided for freeing the image buffers and reinitializing them. Thus, an application can free the buffers, perform its computations, reinitialize the buffers, and then plot the image.

Instances of the display manager residing on the graphics nodes (Figure 2.4) queue buffers to receive incoming images from the computational processes. When the user interface sends a token specifying an image to be displayed, each instance of the display manager decodes the token and takes appropriate action. The process

to which the specified image is transmitted waits (if necessary) for the message(s) carrying the image to complete, then copies the image to video memory. The display manager processes then synchronize and select the board(s) and buffer(s) to display.

In the case of an image that is the composite of images produced on multiple computational nodes, the current implementation sends all of the image segments to a single node, the display manager assembles them in video memory and the image is displayed in the same manner as a single-source image. Meiko has not yet produced software to switch between the "fullscreen" mode (in which a single node is responsible for an entire displayed image) and the "splitscreen" mode (in which each of the graphics nodes contributes a horizontal strip of the image). When such software does become available, the multiple-source images will be divided between (among) the display manager processes in the splitscreen mode.

Because of the issues of dividing images, synchronizing displays, coordinating the efforts of the application, display manager, and user interface, etc., it would be much simpler to implement a system with only a single display manager. A more powerful graphics node (more communication bandwidth, more memory) would enable this change. The ability to combine the functions of the display manager and the user interface into a single process would ease the effort even more. Since the display of images is the least technically demanding of the imaging tasks, we expect that sufficient performance to allow this change should be available in some machines soon.

Video memory in the graphics nodes is laid out such that sequential addresses form horizontal scan lines on the monitor. Seismic data traces are most naturally laid

out in memory on sequential addresses, but are displayed vertically, thus requiring transposition before display. To avoid this extra effort, we have physically turned the monitor on its side (90 degrees clockwise). In this way, left-to-right scanning of memory results in top-to-bottom scan lines—the desired effect. The only drawback of this approach (besides an unknown effect on the monitor) is that the sense of left-to-right is reversed (low memory addresses to the right, high ones to the left, when the opposite would be more natural). This problem is easily solved by reversing the order of the traces as they are copied into video memory. Thus, for a 1K by 1K image, one-thousand transfers of 1 Kb are needed to move the image, rather than a single copy of 1 Mb. This process is still more efficient than transposing a 1K by 1K matrix. Another advantage of this approach is that in splitscreen mode, which divides the screen into horizontal strips, entire traces are sent to a given node, rather than dividing the traces and sending a segment to each node. Dividing the traces would be a much more complex and time-consuming way of producing an image, though it would have the advantage of dividing the load more evenly in the case of an image narrower than the full screen-width.

2.5.3 User Interface/Database

The user interface provides a means of interactively adjusting the seismic imaging parameters, as well as the display and auxiliary parameters. In addition, the user interface provides for control of the system and control over the display of images. As usual, the principle means of accomplishing these tasks will be to provide opaque applications interfaces. One interface takes the form of a simple database, which, besides providing the functionality described in the next section, also allows the state

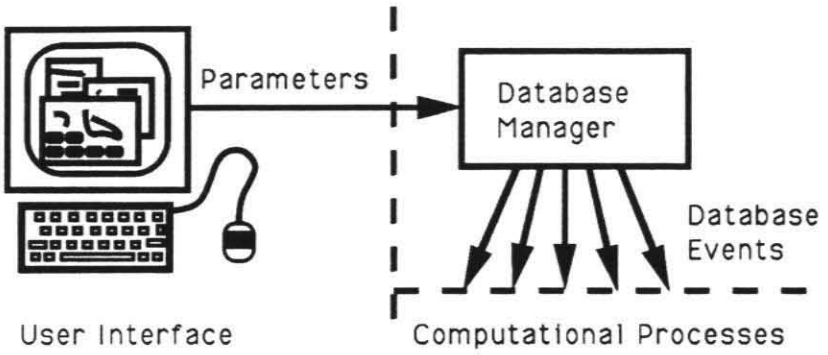

Figure 2.5: The user interface/database.

of the processing to be saved at any time by writing out the database files.

Applications Interface

The user interface itself is left unspecified by the ISIS system software. The application implementor is free to choose whatever means of control is desired. Thus, the interface may be developed as a window-based graphical user interface utilizing keyboard and mouse (as we have done), a simple ascii terminal, or an elaborate visualization system of some type. Regardless of the way the parameters are chosen, modified, and represented internally, they are propagated to the system in only one way: by entry into a parameter database.

The database manager (Figure 2.5) maintains a set of simple, keyed databases. The application opens databases, through a supplied function, as needed, giving each database a unique id, and is returned an opaque handle (much like a file descriptor) for each. The handle is used as an argument to subsequent function calls involv-

ing that database. To store data, the application packs the parameters into byte streams and enters them into the appropriate database. The database manager then propagates the data, by means of database events, to the computational processes. The application-level computational processes are made sensitive to these database events (Section 2.5.4). A change in a parameter, when entered in the database, will trigger the active imaging function if that function has registered an interest in the database. A separate interface is provided for specifying the active function.

We chose to implement the user interface as an X-Windows application using the XView toolkit. X was chosen for its portability and relative standardization. XView was a more difficult choice since it is not the most widely accepted toolkit. XView is, however, supported by Sun, giving us more recourse in the event of problems, and the source is freely available, theoretically making it as portable as any of the other toolkits. In addition, we had available a program called "OpenWindows Developer's Guide" (where "Guide" stands for Graphical User Interface Design Editor). This program simplified the process of creating and modifying the user interface by allowing graphical editing of the windows and controls, and producing source code for the resulting interface. Other programs of this type exist for other toolkits, but were not available at this site. Finally, we preferred XView because it is well documented, and is similar to SunView programming, with which we were familiar.

System Level Implementation

The database manager is a simple database that stores data in buffers and indexes them to unique, user-supplied integer keys. The unique feature of the database is that, when compiled for the host process, it sends a copy of data stored in it to every

other process in the system. The system software in the receiving processes absorb the database messages and enter them into a local copy of the database. Functions are provided for opening and closing databases and inserting, appending, retrieving and deleting entries.

Additional user interface functions are discussed in other sections. Most are implemented as either direct message-passing functions (such as the image display tokens discussed in Section 2.5.2) or they generate special database events (such as the mechanism to set the active function, as discussed in the next section).

2.5.4 Computation/Notifier

The computational processes are the heart of the ISIS system. They respond to user input, provide the seismic processing functions, and produce the images for the user to analyze.

Each instance of a computational process (Figure 2.6b) consists of a set of user-defined, application-level functions, and a controlling function or "notifier." Figure 2.6a is a schematic representation of a typical user-defined function. The user functions are called by the notifier when triggered by a specified database event, as discussed in Section 2.5.3. When a function is called, it is provided with a list of the id's of the databases that have changed since the last call to that function. The application function first retrieves from the databases any new parameters or other needed parameters that did not persist between calls, decodes them, and takes any appropriate action (such as entering the data structures into local storage). Next, the process calls *datarequest()* to request the data necessary to carry out its task. The process then loops over calls to *getdata()* and performs its computations. The

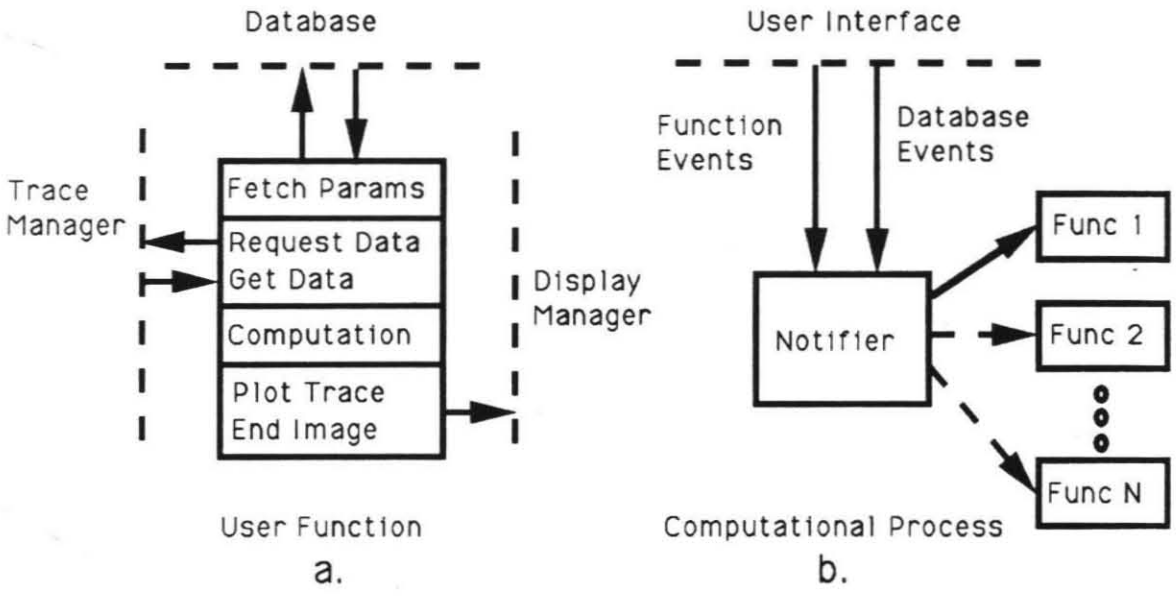

Figure 2.6: An instance of a computational processes: (a) a user-defined function; (b) the controlling process with several user functions in place. The bold arrow running from the notifier to the user function "Func 1" indicates the currently active function.

function may make multiple *datarequest()* calls if a composite image is being formed and multiple gathers are needed. Calls to *draw_trace()* may be interleaved between calls to *datarequest()* or *getdata()* or may all be made when the computations are complete. Finally, *plot_data()* is called and the function returns to the calling process or, alternately, begins the construction of another image. Appendix B contains an example function illustrating the use of these functions.

The notifier (Figure 2.6b) is the main controlling function of the computational process. Each user function is registered with the notifier through a call to *register_func()*. The arguments to *register_func()* are a pointer to the user function, a unique function id number, and a list of the id numbers of the databases that, when changed, cause the function to be called. The user interface specifies the active function through a call to *set_active_func()*, the only argument is the function id number of the desired function. (In the ISIS implementation, the database id's and the function id's are stored as enumerated types in a header file *local.h* common to the user interface and the computational processes.) Only one function may be active at any time, though the user interface may specify that no function be active. After registering all the user functions, the application calls *node_main_loop()*, which does not return until the user interface terminates the system or an unrecoverable error is encountered. Appendix B also contains a fragment of a main program illustrating the use of these functions.

The notifier does little but wait for database events and other incoming messages. When an event arrives, the notifier notes the id and turns the message over to the database software for storage. The notifier then marks the database id on an internal

list of all the functions that have registered an interest in that id. Next, the notifier checks the active function to see if it has registered an interest in the id. If it has not, the notifier waits for the next event. If the function has registered an interest in the database id, the notifier generates a list of id's that have changed since the last call to the active function, clears its internal list for that function, and then calls the function. Each user function takes only two arguments: an array of integers (the list of database id's), and an integer (the number of entries in the list). When the function returns, the notifier checks the return status, if non-zero, the notifier prints an error message and exits, otherwise it again begins waiting for an event.

When the user interface calls *set_active_func()* a function event is generated. The notifier catches this event and resets its own internal notion of the active function and then inspects that function's database list. If any of the function's selected databases have changed since its last call, the function is called immediately, just as if a database event had been received. Otherwise the notifier waits for the next event.

Not all database events are directed toward user functions. Some database id's are reserved for the system. When these events arrive, the notifier takes direct action on them. For instance, the function events generated by *set_active_func()* are simply database events with a particular id reserved for that purpose. Another database id is used only to signal the notifier that the user interface wants it to exit.

Some readers will note that this notification based style of programming is not original. It is, in fact, very similar to the SunView/XView model of programming—so much so that some of the terminology is the same. This coincidence is not accidental,

the instant model was, indeed, inspired by that approach.

Through the use of the database manager we have solved a number of problems. First, it provides the obvious service of propagating the processing parameters throughout the system. Second, the system software operates without any knowledge of the contents of the database entries. The size and content of those entries is entirely under the control of the application. The contents need be known only in the user interface and the processing functions. These functions perform the packing and unpacking of the data, and are in a position to do so since they are both user-defined. Thus, parameter data structures can be added, removed, and changed without recompiling the system software. Third, the database manager/notifier system provides the user with a means of controlling the behavior of the processes without the user interface programmer ever having to resort to parallel programming or explicit message passing. Another advantage is that data representation issues may be handled at the same time the data is packed and unpacked. Meiko supports Sun's XDR package and we chose to use XDR for this purpose. The XDR package may not be as efficient as custom software and is a bit awkward to use, but it is general purpose, supported by both vendors, and was available (thus cutting our development time).

Note that through the use of the trace manager, display manager, and database manager, the parallelism of those parts of the machine are completely hidden from the application. The only level of parallelism that the application programmer needs to consider is that of the computation itself. To facilitate parallelization, each process is supplied with the total number of computational processes, as well as its own

position within that total. Most seismic processing algorithms are relatively easy to decompose by either data, or domain, or both. The trace manager provides the mechanism to perform data decomposition, and the display manager provides a means of assembling images from domain decomposed processing functions. To make possible more complicated algorithms, generic functions are provided for sending and receiving messages to and from computational processes and to and from the user interface.

The database manager sends database events to all of the processes in the system, not just the computational processes. This action is to allow for a uniform treatment of parameters throughout the system. In this implementation, the trace manager and the display manager simply discard any database events they receive. But in other implementations those processes may need access to system parameters. For instance, the graphics nodes may need access to the structures containing the plot scaling information, and the I/O nodes may require access to the velocity model in order to apply NMO correction. In this implementation, however, those functions are performed on the computational nodes, where the parameters are already available.

2.6 Current Status

We have implemented the system software, as described in the previous sections, as well as a number of seismic processing functions to establish the prototype system. As mentioned above, the user interface is implemented as a window-based graphical interface that accepts keyboard and mouse input. We attempted to make the user interface intuitive so that it would be easy to learn and use. Much of the basic

functionality described in Chapter 1 has been implemented. We can display movies of shot, midpoint, and receiver gathers, edit traces and set mutes. We have also established a means of performing interactive velocity analysis.

CMP stacking is performed in parallel by dividing the reconstruction line into N segments (where N is the number of computational processors). Each processor, i, selects the i^{th} line segment, and stacks the CMP gathers within that segment. While the software to perform interactive focusing is fully enabled, the machine's performance is not yet high enough to make the process interactive on anything but very small segments or small data sets. The addition of more nodes would facilitate this functionality.

We have also implemented three migrations, all of the Kirchhoff summation type, as described by Schneider [34]. We have implemented two varieties of post-stack migration. The first, a simple Kirchhoff time migration, assumes straight rays, but allows for lateral velocity variations. The second is again a Kirchhoff method, but is a depth migration that allows for curved rays and lateral velocity variation by explicit travel time computations. The travel times are calculated by a finite difference solution to the Eikonal equation (see Vidale [38]). Amplitudes are scaled assuming straight rays, however. The scaling is proportional to the cosine of the angle between the vertical axis and a line connecting the image point and midpoint, and inversely proportional to the distance between those points.

Segments of the stack, as we have mentioned, are divided among the nodes. The post-stack migrations are designed to make use of pre-existing stacks. The time migration divides the image plane among the nodes—each node works with

a segment of the image plane that corresponds to its segment of the stack. The migration function calculates the contribution to the image of the local portion of the stack, and then passes the image on to the next-higher numbered node (the highest-numbered node passes the image to the lowest-numbered). The image segments are passed around the loop until each process receives its original image back, at which point the contribution of the entire stack has been added. At that point the migration is done and the image is plotted.

The travel time migration uses a slightly different method of forming a composite image from the distributed input. Each process maintains an entire image plane to which it adds the contribution of its portion of the stack. When finished, the image is divided into N segments (where N is the number of computational processes). Each segment is transmitted to its corresponding node (i.e., the i^{th} segment is sent to the i^{th} node), the segment corresponding to the local node is kept. Each processor receives $N-1$ image segments from the other processes. These segments are summed into the local segment and are then plotted.

We have also implemented a prestack Kirchhoff migration. This migration also uses explicit travel time calculations and, again, assumes straight rays for scaling the amplitudes. The prestack migration is particularly efficient for parallel implementation; none of the significant computations are performed redundantly.

To perform the prestack migration, the output image plane is divided into N vertical strips, where N is the number of computational processes. The velocity model is mapped onto the data plane (i.e., a plane formed by a line on the surface long enough to encompass the source and receiver locations of all the data traces

used in the migration, and a depth equal to the depth of the image plane). For each point in the image plane, the corresponding point in the velocity plane is treated as a point source, and travel times are computed to the surface (i.e., to the positions of the sources and receivers). The travel times corresponding to the source and receiver positions of a given trace are summed, yielding the travel time from the seismic source to the image point and then to the receiver. The value of the datum recorded at that time in the trace is scaled and summed into the image plane at the current image point. To avoid redundant travel time calculations, we compute the travel times once for each image point and store the travel time solutions corresponding to the surface positions. Thus, we loop over each trace in the survey, and for each trace we loop over each point in the image plane. Each processor must have access to the entire data volume, but the process is compute bound, so there is no need to attempt any optimizations in the trace manager.

Memory constraints, however, prevented us from being able to store surface travel times for every point in the image plane. One solution would be to make multiple passes through the data. Instead, we chose to compute travel times on a coarser grid and interpolate to the values in-between. Note that this cuts down the number of passes through the data, but does not necessarily eliminate multiple passes altogether. Tests showed that this implementation was accurate enough to correctly migrate the data. The substantial savings in computer time from reducing the passes through the data, reducing the size of the travel time computation, and reducing the total number of travel time computations that needed to be performed was offset by the expense of interpolating the travel times. The approximate time for a prestack

migration of the Hosgri data (see Chapter 3) ranged from about six hours for a large image, down to about 1.5 hours for the smaller ones.

Neither of the post-stack migrations was fast enough to be interactive, requiring 5–10 minutes to migrate a stack of the Hosgri data. As with the stacking, the migrations are designed to be interactive if the performance of the hardware and software can be upgraded sufficiently.

The functions mentioned above provide enough of a core processing system to perform basic seismic imaging. In Chapter 3 we discuss the application of the system to a marine data set. This work allowed us to apply the system to a significant geological problem, and to evaluate the imaging system.

2.7 Future Work

Considerable work remains to done on ISIS. From an applications point of view there are an almost endless number of seismic processing functions that may be added to a system such as this, but there are a few basic ones that should be given special consideration. (Some of these functions are discussed in more detail in Chapter 3.) Making the system more usable as an end-to-end processing system would require the addition of modules for interactive filtering (both frequency and F-K) and deconvolution. Marine data often requires multiple suppression, and land data usually requires statics corrections. Both of these functions would be present in a complete system. In addition, more sophisticated velocity analysis would be helpful in some situations. Finally, the current system works exclusively with NMO velocities, but it would be useful to manipulate the velocity model in either NMO or interval velocity

format.

It is always desirable to have faster processing, especially on an interactive system such as ISIS. Improving the algorithms and implementation could help in some instances, but in general, faster performance must come from better compilers or by hand coding the most demanding functions. The i860 is still young, as is its compiler technology; we are hopeful that significant performance advances can be made without resorting to programming in assembly language.

Some portions of the ISIS system software also need improvement. A general rewrite of some portions of the code to isolate system dependencies and improve modularity and efficiency would be useful. We would also like to implement a better system for error handling and recovery, as well as more user-friendly diagnostic messages.

As mentioned in Section 2.5.1, the performance of the trace manager could be improved by a more intelligent data management scheme. Enhancements to the trace manager have the potential to greatly improve the ability of the system to support truly interactive processes.

2.8 Conclusions

We have discussed each of the four main divisions of ISIS: the trace and display managers, the computational processes, and the user interface. One of the design goals of each of the parts was to provide a simple interface that hides the parallel implementation from the applications programmer while still providing the performance offered by the parallel hardware. The trace manager works in a concurrent, multi-process

multi-disk environment, but the application sees only sequential-looking functions. The display manager provides another equally simple interface to an otherwise complicated hardware system. The database manager allows for an interchangeable front-end interface, as well as a simplified means of controlling the processing. The notifier allows for easy addition of processing functions, and changes in parameters.

Chapter 3

Application

3.1 Introduction

The previous chapters have presented the concept and implementation of an interactive seismic imaging system. In this chapter we discuss the application of the system to a particular data set; part of a reflection profile known as EDGE line RU-3. The line was shot in 1986 by the EDGE continental margin consortium (see Mooney [29]). The survey was conducted offshore central California (Figure 3.1) to assess the tectonic history of the area. We selected a subset of line RU-3 that crosses the controversial Hosgri fault zone. First we provide background information and discuss the main issues surrounding the Hosgri fault. We then discuss the data and processing history, followed by a discussion of our results and findings of both the geological problem and of the imaging system.

3.2 Background

The Hosgri fault zone, located offshore central California (Figure 3.1), though barely

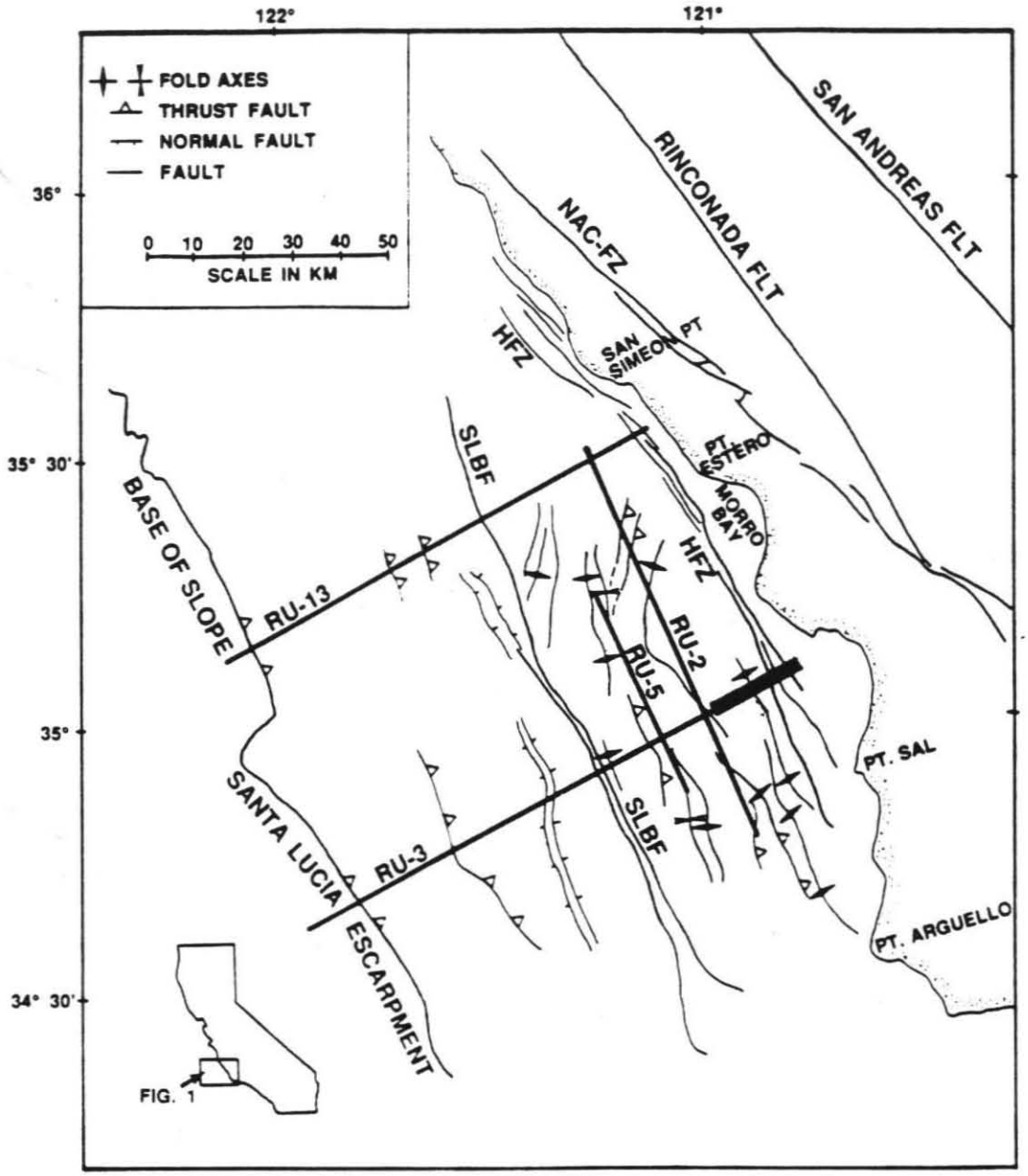

Figure 3.1: Map showing offshore central California. Shown are the Hosgri fault zone (HFZ), the Santa Lucia Bank fault (SLBF), and the EDGE seismic lines (RU-2, 3, 5, 13). The bold line at the end of RU-3 is the portion of the data analyzed here. (After Meltzer and Levander [27].)

recognized two decades ago, is the source of considerable controversy. The exact nature of the fault, its deformational history, its Holocene activity, and its potential for producing a large earthquake are the subjects of extended debate. The Hosgri itself is approximately 150 km long, is offshore for its entire length, and is made up of multiple anastamosing strands. Two principle schools of thought have developed over the deformation associated with the Hosgri fault—those favoring primarily right-lateral strike-slip, and those favoring an east-up thrust displacement.

Atwater [1] presented a model of the Cenozoic tectonic history of western North America in which a mid-Tertiary trench existed off the California coast where the Farallon plate was subducting under the North American plate. As the continent overrode the spreading center between the Pacfic and Farallon plates, the Pacific and North American plates came into contact, about 21–29 m.y. b.p., and the San Andreas fault was formed to absorb the differential motion between the plates. Minster and Jordan [28] derive a plate interaction model that suggests right-lateral deformation west of the San Andreas is occurring at 6–25 mm/yr parallel to the fault and 4–13 mm/yr crustal shortening orthogonal to the fault. More recently, DeMets et al. [6] revised the estimate of unaccounted-for relative motion between the Pacific and North American plates to be approximately 5 mm/yr parallel and 7 mm/yr perpendicular to the San Andreas fault. Thus, structures west of the San Andreas fault, such as the Hosgri, may be absorbing strike-slip or compressional deformation, or both.

Hoskins and Griffiths [21] were the first researchers known to have produced a map showing the then unnamed "Hosgri" fault. While not discussed in the paper,

the fault appears on their cross section as a high-angle fault forming the eastern boundary of the offshore Santa Maria basin, separating Tertiary sediments from Franciscan rocks to the east. H. C. Wagner [39] named the fault the "Hosgri" after Hoskins and Griffiths.

Wagner's work, and several other studies in the 1970's, were commissioned or motivated by Pacific Gas and Electric Company (PG&E) in relation to the licensing of the Diablo Canyon Nuclear Power Plant. The early findings of this work suggested that the Hosgri was part of an extensional regime and had also facilitated some unspecified amount of strike-slip displacement (PG&E [10]).

Early studies concluded that considerable strike-slip displacement has occurred along the Hosgri. C. A. Hall [15,16] suggested that the Lompoc-Santa Maria basin formed as a Tertiary pull-apart basin, beginning about 14 m.y. b.p., in the late Miocene. Hall states that during approximately the last 5 m.y., the western (i.e., offshore) part of the basin has been displaced 80 to 95 km northwest along the Hosgri fault. These figures give a displacement rate, on average, of about 16 mm/yr.

Hall [15] also argues that the San Simeon Fault to the north of the Hosgri (Figure 3.1) is part of the same system as the Hosgri. Similarly, Silver [36] argues that the displacement on the San Gregorio-Sur fault system to the north (Figure 3.2) is likely continuous with the San Simeon-Hosgri fault zone, instead of heading inland on the Palo Colorado fault south of Monterey, as had been previously suggested. The San Gregorio fault is generally accepted to be a near-vertical right lateral strike-slip fault (e.g., Saleeby [33]).

Graham and Dickinson [13,14] also argue against the notion that the Palo Col-

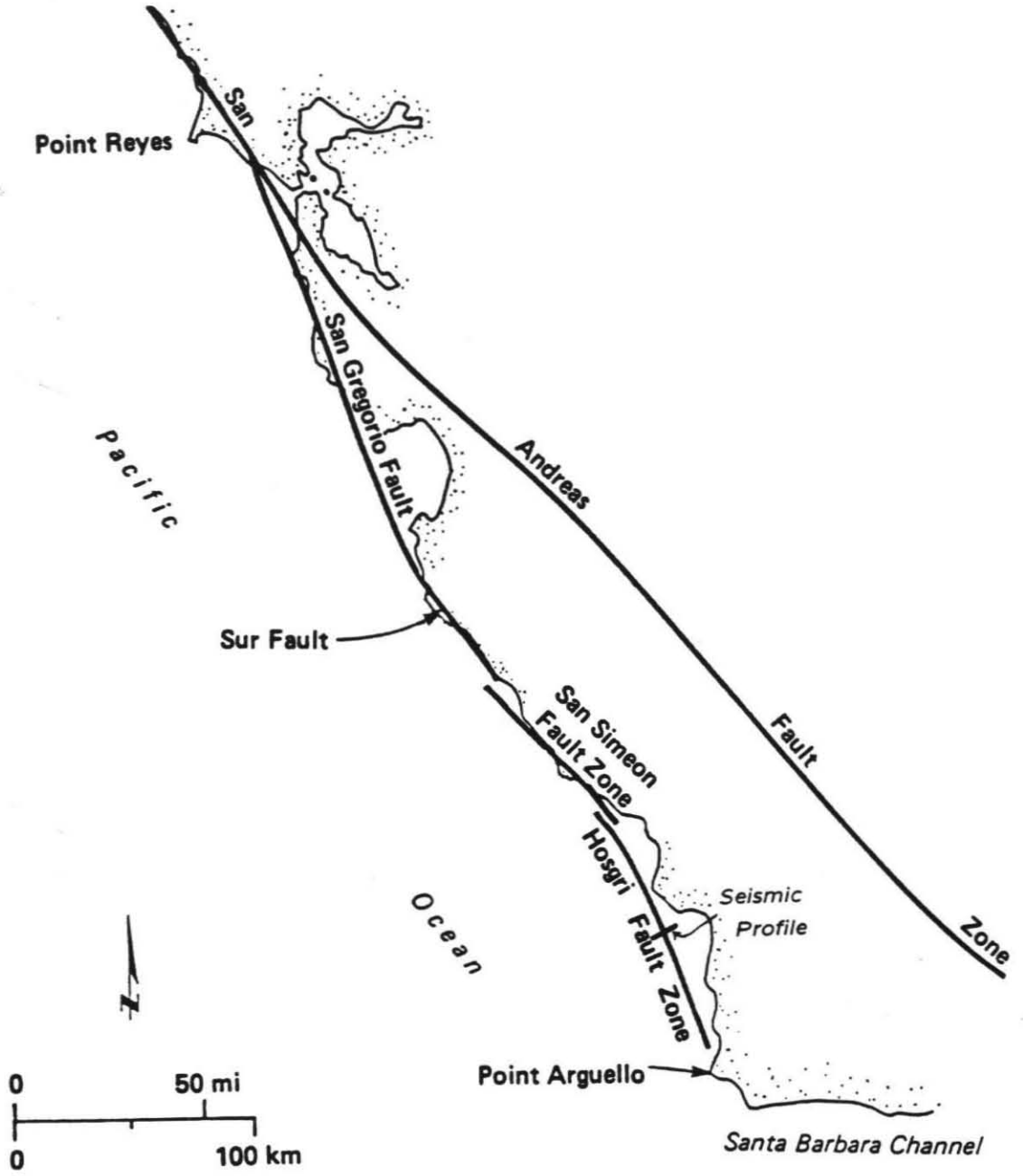

Figure 3.2: Map showing the major fault systems of offshore central California. The position of the seismic profile is indicated. (After PG&E [10].)

orado fault is strike-slip, and also suggest that the San Gregorio and Hosgri form a continuous system. They go on to argue for 115 km of Neogene right lateral displacement on the San Gregorio-Hosgri system. Their studies indicate that displacement on the combined system initiated in mid- to late-Miocene and continued for at least 2 million years. They also suggest that Holocene displacement, while detectable, is minor.

Other workers have also found significant displacement on the San Gregorio fault. Weber and Lajoie [40] found from the study of offset geologic formations on two onshore segments, that the San Gregorio Fault has experienced approximately 6.3 to 13.0 mm/yr of right-lateral displacement over the last 200,000 years (with the total motion accumulating on several fault strands) and some minimal amount of vertical displacement as well. J. Clark et al. [3] working with similar methods on the Reyes Peninsula, north of San Francisco, discerned about 150 km of right lateral slip on the San Gregorio since late Miocene, continuing to the present day (approximately 12.5 mm/yr). Coppersmith and Griggs [4] find field evidence for late Pleistocene and Holocene offset on the San Gregorio, as well as recent seismicity due to regional right shear and compression. Their field investigations show no signs of recent fault creep, however.

Using earthquake records, Gawthrop [11] found the entire area to be seismically active and associated the seismicity with northwest trending faults (i.e., the San Gregorio, San Simeon, and Hosgri). He suggests that the system could be accumulating strain at a rate of 16 mm/yr along its entire length. If this strain is released by a rupture spanning the entire length of the system, the combined system could yield an

earthquake with a magnitude of up to 8.0, and a repeat time of 250 years. Farther to the south, on the San Simeon fault, Hanson et al. [19] find significant ($3.9^{+4.5}_{-2.4}$ mm/yr) right lateral slip during the late Pleistocene from San Simeon Point to Ragged Point based on the study of marine terraces.

Much of the work on the San Gregorio-San Simeon fault system is extrapolated to include the Hosgri, since little geologic work can be directly performed on that fault. Most of the studies cited above involves field relations to the north of the Hosgri, coupled with gravity and magnetic work, and some high-resolution seismic profiles. A geological study on the Hosgri itself, by Hamilton and Willingham [18] based on stratigraphic relationships and well data, implies a maximum of 10 to 20 km of right lateral slip along the Hosgri fault since the Miocene. It is impossible to thoroughly assess this work since it appeared only as an abstract, however, it is difficult for those who argue for greater amounts of slip to explain the reason the Hosgri seems to simply die out near Point Arguello (Figure 3.1). If Hamilton and Willingham's conclusions prove true, a great deal of earlier work will need to be reinterpreted.

Hamilton [17] also asserts that the San Gregorio-Hosgri system branches just south of the Monterey Bay into inland-trending strike-slip and southward-trending dip-slip components. While Hamilton accepts that the San Gregorio fault itself exhibits substantial Quaternary slip, he finds that south of Point San Luis the Hosgri has essentially no seafloor expression and, though it continues, appears to exhibit little slip (a few mm/yr). North of Point San Luis the fault is marked by minor scarps and sags on seismic profiles.

Up till now we have discussed the arguments for large-scale right-lateral displace-

ments on the San-Gregorio fault system and, implicitly, the Hosgri. As mentioned earlier, however, this interpretation is by no means the only one. In the early to mid-1980's many researchers argued that the Hosgri was a thrust or reverse fault responding to the regional compressive stress. Much of this shift in opinion was due to increased coverage of the fault zone by seismic reflection profiles.

Crouch et al. [5] argue that the offshore faulting is dominated by thrust and high-angle reverse faulting that flattens to become thrusts at depth. They point to several lines of geologic evidence that, they argue, indicate that much of the faulting associated with the San Gregorio-Hosgri system is thrust related and that the strike-slip component is largely overstated. The central conclusion of their work is that compression plays an important role in the development of the region—as much as 30 to 70 km of shortening in the past 5.5 m.y. They suggest that there exists a crustal detachment at approximately 12 km depth, on which the compression is being transmitted across the region.

Based on the geology and tectonic setting, McCulloch [25] characterizes the structures in the offshore Santa Maria basin as extensional or transtensional with a compressional overprint. He suggests that the Santa Maria basin formed through rapid subsidence that was controlled by extensional wrenching in the Oligocene through mid-Miocene. McCulloch argues that this period was followed by compressional deformation in the central and seaward portions of the basin in the Pliocene epoch or later.

Namson and Davis [31] constructed geological cross sections across central California. Their section shows the Hosgri as a major thrust fault largely responsible for

the anticlinal deformation to the east, and the uplift of the Coast Ranges. The regional stress field has components of both compression normal to, and shear parallel to, the regional fault systems. Namson and Davis suggest that the thrust faulting resulting from regional compression is largely detached from the right lateral strike-slip faulting resulting from regional shear. This idea found considerable support in the work of Zoback [42] and Mount and Suppe [30]. These studies of borehole breakouts, in situ stress measurements, heat flow, focal mechanisms, etc. indicate that the San Andreas fault is a low friction zone that effectively decouples the shear and compressional forces of the Pacific/North American relative plate motion. These studies find that the shear stress is on the order of 10 to 20 MPa, the deviatoric stress is on the order of 100 MPa. This evidence tends to refute the idea of wrench tectonics, at least along the San Andreas and perhaps along the Hosgri fault as well.

As does Crouch [5], Namson and Davis [31] suggest a regional detachment surface at 10 to 12 km depth. This suggestion finds support in deep seismic reflection work. Ewing and Talwani [9] infer subducted oceanic crust at about 6 seconds depth under much of the region from the continental slope to the Santa Maria basin. Meltzer and Levander [27] discuss similar findings, and suggest that the top of the subducted crust may form the mid-crustal detachment.

Later reflection studies of the Hosgri fault zone have also found a significant thrust component on the Hosgri. McIntosh and others [26], using data from EDGE line RU-3, suggest initial normal or strike-slip faulting, followed by thrusting. In various locations in the Santa Maria basin they see compressional features formed in the last 3–5 m.y. on earlier extensional faults. Unlike McCulloch [25], McIntosh et al.

find that the Hosgri was originally an extensional or transform fault. They suggest that the fault has been subsequently reactivated as a thrust or reverse fault—a displacement pattern that they argue continues to the current day. McIntosh et al. see both high- and low-angle fault strands, but are unable to determine which strand cuts the other.

Utilizing a subset of the same reflection data used by McIntosh et al. [26], Louie [23] studied diffraction hyperbolas and performed prestack migration to image fault plane reflectors in the Hosgri fault zone. He estimated that these reflectors must have a dip of 30°–50° E. He also finds that sedimentary reflectors truncate at the location of the imaged fault plane, and argues that thrust controlled the deposition of the basin sediments.

Other studies, however, have cast serious doubt on the notion of substantial thrust having occurred on the Hosgri fault. D. Clark and others [2] using methods similar to those of Namson and Davis [31,32], argue that the Queenie structure in the central offshore Santa Maria Basin formed during a brief period of NE-SW directed shortening between 3 and 5 Ma (at the time of the onset of Pacific-North American plate compression). Since that time, however, they argue that the shortening has stopped or slowed to no more than 0.005 mm/year. Based on the seismicity, the small amount of folding east or west of the Queenie structure, and their interpretation of the Queenie as a fault-bend fold, they see little evidence for significant shortening along a thrust or detachment fault beneath the Queenie structure.

Meltzer and Levander [27] using reflection data from EDGE line RU-13 to the north of line RU-3 (Figure 3.1), infer that the Santa Maria Basin exhibits two periods

of Neogene deformation, an early extensional/strike-slip phase in mid-Miocene, and an upper Miocene-lower Pliocene compressional phase. They do not detect deformation of the upper two (Pliocene) units, and they observe only 1–3 km of shortening of Neogene sediments, thus any significant shortening due to Pacific-North American plate motion extends east of the basin and is distributed across the entire continental margin. This assertion is supported by the work of Namson and Davis [31] who find active compressional folds both west and east of the San Andreas fault.

Later work by Namson and Davis [32] reverses their earlier paper's contention that the Hosgri plays a major role in the tectonics of the area [31]. The later work still places the Hosgri at the western edge of the Point San Luis Anticline but they see little recent slip on the fault itself, and argue that the anticline has far more structural relief than can be attributed to the measurable displacement on the Hosgri. They agree with other workers that the Hosgri was a high-angle fault responsible for Miocene and Pliocene sedimentary thickening on the downthrown block. Later, they suggest, the growth of the Point San Luis Anticline deformed the fault from steep-west to steep-east dip. The fault has since experienced minor reactivation as a reverse fault, probably due to stress from the formation of the Point San Luis Anticline.

A re-examination of the paper by Crouch et al. [5] also leads one away from the conclusion of significant thrust on the Hosgri. While the majority of the data used are not presented, Crouch et al. 's "interpretive line drawings" do not show a significant amount of thrust on the Hosgri—certainly not enough to make a significant contribution to the 30 to 70 km of shortening across the margin, for which they argue. There is also no evidence of active thrusting in any of their figures. (Crouch et al.

suggest that the faulting ceased by mid- to late-Pliocene.)

Snyder [37] suggests that the low-angle throughgoing faults of Crouch et al. may instead be local ramp structures in the Monterey formation in which a rigid layer breaks and thrusts over itself while the overlying ductile layers fold. Folds of this type have been observed in the field. This explanation would seem to agree with the small amount of thrust seen on the Crouch profiles, and refute the notion of throughgoing thrust faults.

Pacific Gas and Electric Company [10] reviewed virtually all of the available literature, as well as commissioned several studies and purchased proprietary data on the area, in conjunction with the licensing of the Diablo Canyon Power Plant. The result of this extensive investigation was the conclusion that the northern three-fifths of the Hosgri fault is characterized by late Quaternary right-lateral strike-slip of approximately 1–3 mm/yr. The slip decreases to the south, possibly becoming zero south of Point Sal. They do not observe any evidence for late Quaternary activity on any low-angle thrust faults.

We are thus left with a difficult set of seemingly conflicting results: 1) Studies have shown substantial amounts of late Cenozoic slip on the San Gregorio and San Simeon faults, 2) these faults have been argued to be continuous with the Hosgri, 3) the Hosgri itself shows little evidence of major recent strike-slip deformation, 4) the Hosgri seems to die out to the south, suggesting that the slip is non-uniform, 5) the Hosgri does not show any significant recent thrust component.

Hornafius [20], and Luyendyk and Hornafius [24] provide a means of reconciling many of the apparent discrepancies in the data, however. Based on paleomagnetic

and stratigraphic studies, they propose a clockwise rotation of the Santa Ynez Mountain Range (Figure 3.3) of 95° ± 9° in two stages during the Late Miocene/early Pliocene and again in the Plio-Pleistocene. This rotation resulted in shear along a series of parallel NW-SE trending faults. The initial rotation resulted in the opening of the onshore and offshore Santa Maria basins as well as strike-slip motion on the Hosgri. The second rotation resulted in strike-slip displacement of the offshore basin to the north as well as NE-SW directed compression within the onshore basin. Because the second rotation was about a different pole than the first, the resultant compression in the onshore Santa Maria Basin resulted in lesser slip on the southern portion of the Hosgri than on the northern. The total slip on the Hosgri from both episodes was a maximum of 140 km, which fits well with other estimates of slip.

The seismicity of the area lends support to this model. Eaton [8] determined focal mechanisms for several moderate (~M5.0) earthquakes along the coast of central California (Figure 3.4. The focal mechanisms progress from left lateral reverse oblique slip to the south (near Santa Barbara and in the onshore Santa Maria basin), to simple reverse near Point Sal, to right lateral reverse oblique near San Simeon, to right lateral in the north (near Pinion Head and Point Sur). The Point Sal earthquake occurred offshore, but to the east of the Hosgri. Both of the fault planes of the Point Sal solution intersect the Hosgri fault at an angle. They are a much closer fit, however, to either the Lion's Head Fault or the Orcutt Frontal Fault, both of which splay off the Hosgri and head inland in the area of the quake (see Figure 3.3).

Namson and Davis [32] argue for significant shortening across the continental margin. Through the use of balanced cross sections they find 6.7–13.4 mm/yr con-

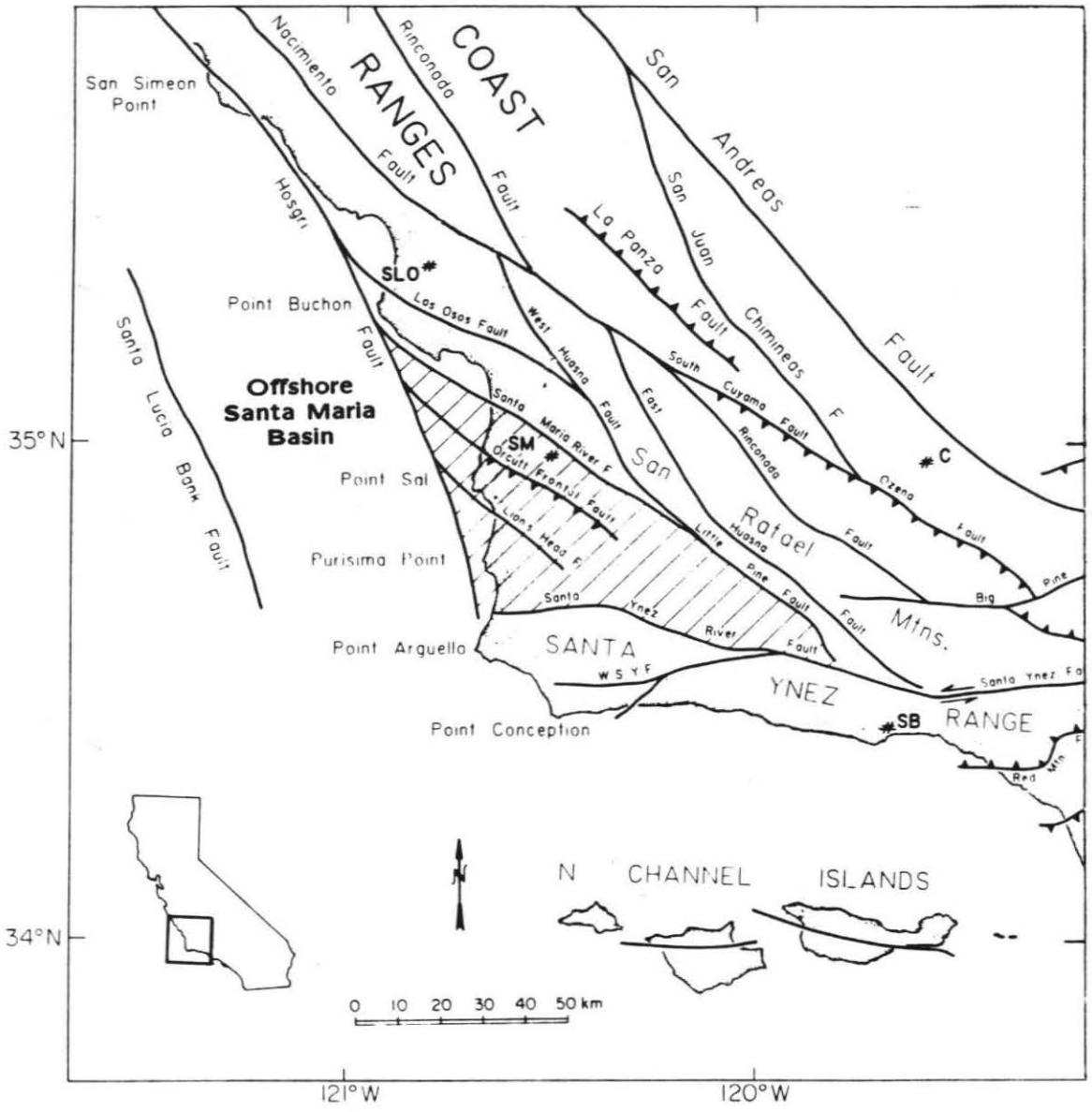

Figure 3.3: Map of west-central California. The onshore Santa Maria basin is the cross-hatched, nearly-triangular region in the center of the figure. Cities shown include Santa Barbara (SB), Santa Maria (SM), San Luis Obispo (SLO), and Cuyama (C). (After Hornafius [20].)

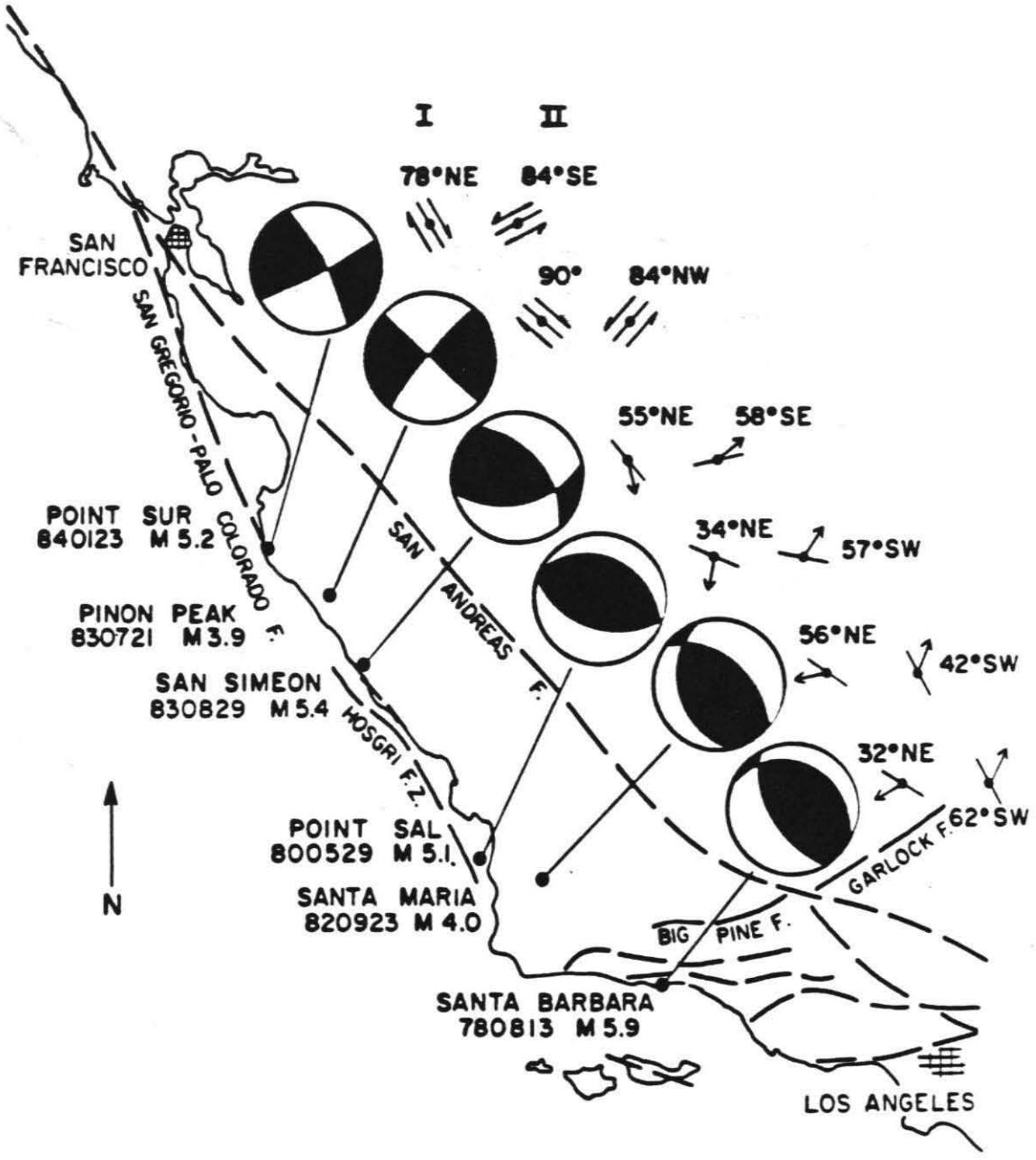

Figure 3.4: Map showing earthquake focal mechanisms for selected earthquakes in west-central California. (From Eaton [8].)

vergence in the entire region since 2–4 Ma, with 1.8–3.6 mm/yr divided among the offshore structures. Their section suggests a total of 7.2 km of slip transferred to the offshore Santa Maria basin and continental margin to form convergent structures there. Their section stops in the central Santa Maria Basin, however, so it is difficult to assess whether or not the structures there show sufficient convergence to account for the slip. As mentioned above, Clarke et al. [2] and Meltzer and Levander [27] suggest not.

If the Hornafius/Luyendyk [20,24] model is correct, it conflicts with on the total shortening calculated by Namson and Davis [31,32]. Their balanced cross sections assume that the deformation is primarily compressional and extensional, and that there is little out-of-plane strike-slip deformation. Though they are not convinced of the validity of significant post-Miocene strike-slip on faults in the area, it is unknown if they could adjust their model to account for such displacements.

It may not be necessary to postulate significant offshore shortening, however. The total San Andreas fault-normal shortening due to the relative motion of the Pacific and North American plates predicted by DeMets and others [6] is approximately 7 mm/yr. Namson and Davis (1990) find 6.7–13.4 mm/yr of convergence west of the San Andreas, while in the earlier paper (Namson and Davis 1988) they discuss an actively developing fold and thrust belt east of the San Andreas that is accumulating late-Cenezoic shortening at a rate of about 2.75 mm/yr. Thus, even their lower bound on velocity west of the San Andreas may be overstated, and could be reduced by the entire 1.8 to 3.6 mm/yr that they transfer offshore. It is thus possible, and even likely that the offshore region is characterized by compressional stress, but that stress

does not result in significant deformation. Because the rotation of the Santa Ynez Range may be continuing, we can expect some minor deformation in the region of the Hosgri fault. That deformation may take the form of right lateral strike-slip, thrust or reverse faulting, or a combination of the two.

There is some evidence for minor recent deformation on the fault. Both Wagner's [39] and PG&E's [10] seismic profiles show some apparent fault scarps on the ocean floor in the areas to the north of Point San Luis. South of that area, there is little apparent seafloor relief at the mapped trace of the fault.

There remains, then, considerable debate as to the exact nature of the Hosgri fault, its extent, and its potential for significant seismicity. We elected to reprocess a portion of the seismic data in order to image the near-fault structure, with the hope of characterizing the fault and understanding its history.

3.3 Processing History

As mentioned above, the data were acquired as part of the EDGE consortium's continental margin program. This study uses the easternmost 182 shots of EDGE line RU-3 (Figure 3.1). The acquisition equipment consisted of a 180 channel hydrophone array, with group spacing of 25 m, and a maximum offset of approximately 4700 m. The source was an air gun array fired at 50 m intervals (yielding 12.5 m midpoint spacing and 45-fold gathers). Sixteen seconds of data were recorded at a 4 ms sample rate. Little energy was apparent beyond approximately 40–45 Hz, we therefore band-pass filtered the data with a 6–50 Hz filter and then resampled at 8 ms. Since the purpose of this investigation was to study upper crustal structure, we only preserved

the first six seconds of data.

The survey extends only about 2.5 km east of the mapped trace of the Hosgri fault, due to the encroachment of state waters. Because of this geometry, the CMP gathers begin to roll out at approximately the fault location, and the far-offset traces become unavailable. As will be discussed repeatedly below, the lack of far-offset traces diminishes the effectiveness of a number of seismic processing techniques. Additional coverage might greatly improve the results.

3.3.1 Preprocessing

Because the ISIS system does not yet have F-K filtering, multiple suppression, or deconvolution modules, some preprocessing was performed on the data in a non-interactive manner. Our usual approach was to load the data onto ISIS, perform several processing steps to determine the necessary preprocessing, and then perform those tasks in a standard manner. We were able to keep multiple data sets on disk in the ISIS system, which enabled us to compare the results of the various processing steps with each other and with the raw data. This "manual" (i.e., non-interactive) processing gave us an added appreciation for the advantages of interactive processing.

Early stacks of the data showed two significant problems. A considerable amount of low-velocity, linear coherent noise contaminated the section, especially on the east end of the line (Figure 3.5). The other problem was a number of significant water-bottom multiples (Figure 3.6).

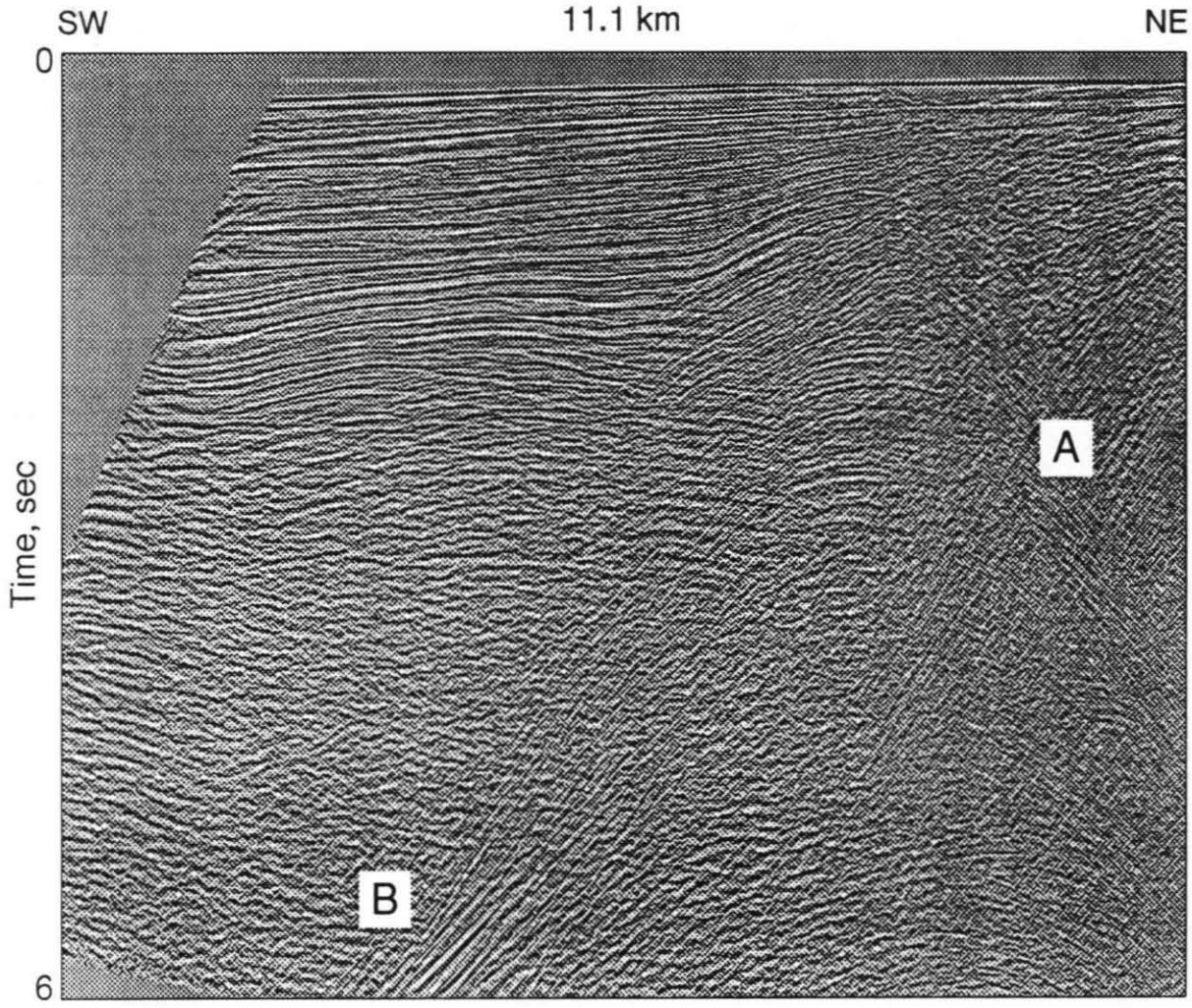

Figure 3.5: A CMP stack of the raw Hosgri data. Note the linear coherent noise (A), and the steeply dipping energy late in the section (B). Automatic gain control (AGC) has been applied to bring out the faint details.

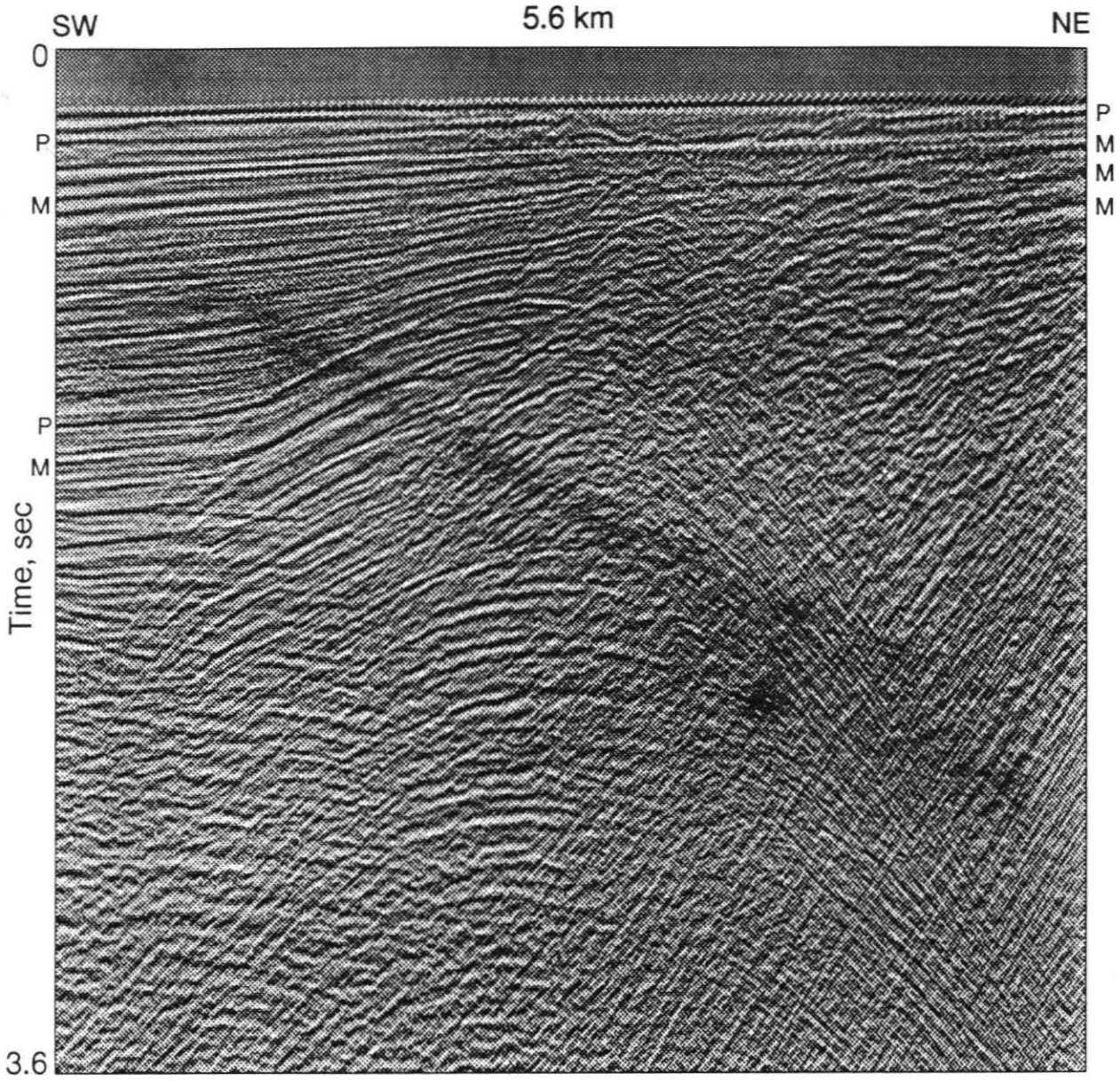

Figure 3.6: Detail of the Hosgri CMP stack (Figure 3.5). This figure is enlarged by a factor of two over Figure 3.5. Note the primary reflections (P), and their associated water-bottom multiples (M). Automatic gain control (AGC) has been applied to bring out the faint details.

Coherent Noise

The coherent noise seen in Figure 3.5 is a common feature of marine seismic profiles (see, e.g., Larner et al. [22]). This noise is generally the result of side-scattered energy from the water bottom or sub-bottom. The most effective means of dealing with it is to apply an F-K filter to the shot and/or receiver gathers in order to remove the noise prior to other processing. This approach not only improves the appearance of the stack, but also aids the velocity analysis by removing spurious hyperbolic arrivals in the CMP gathers.

While the noise does not seem to seriously impair our ability to interpret the stack, it appears strong enough at the east end of the survey to affect the velocity analysis, and the noise could contaminate post-stack migrations. We therefore applied F-K filtering to the shot gathers in an attempt to suppress the noise. We tested a variety of filter settings before choosing a set to apply to the entire data volume.

The F-K filtering was reasonably successful in removing the coherent noise from much of the stack (Figure 3.7), however, the midpoint gathers east of the fault were still seriously contaminated by low-velocity, hyperbolic arrivals. These arrivals may be low incident-angle diffractions from the vicinity of the fault and diffractions from a rough, sub-seafloor reflector (discussed later). Because of their low apparent velocity, the multiple suppression techniques discussed in the next section had some influence in eliminating these arrivals but, as with the multiples, this effort is undercut by the lack of far-offset traces. The F-K filtering, while effective, did not expose any previously-hidden reflectors or structure.

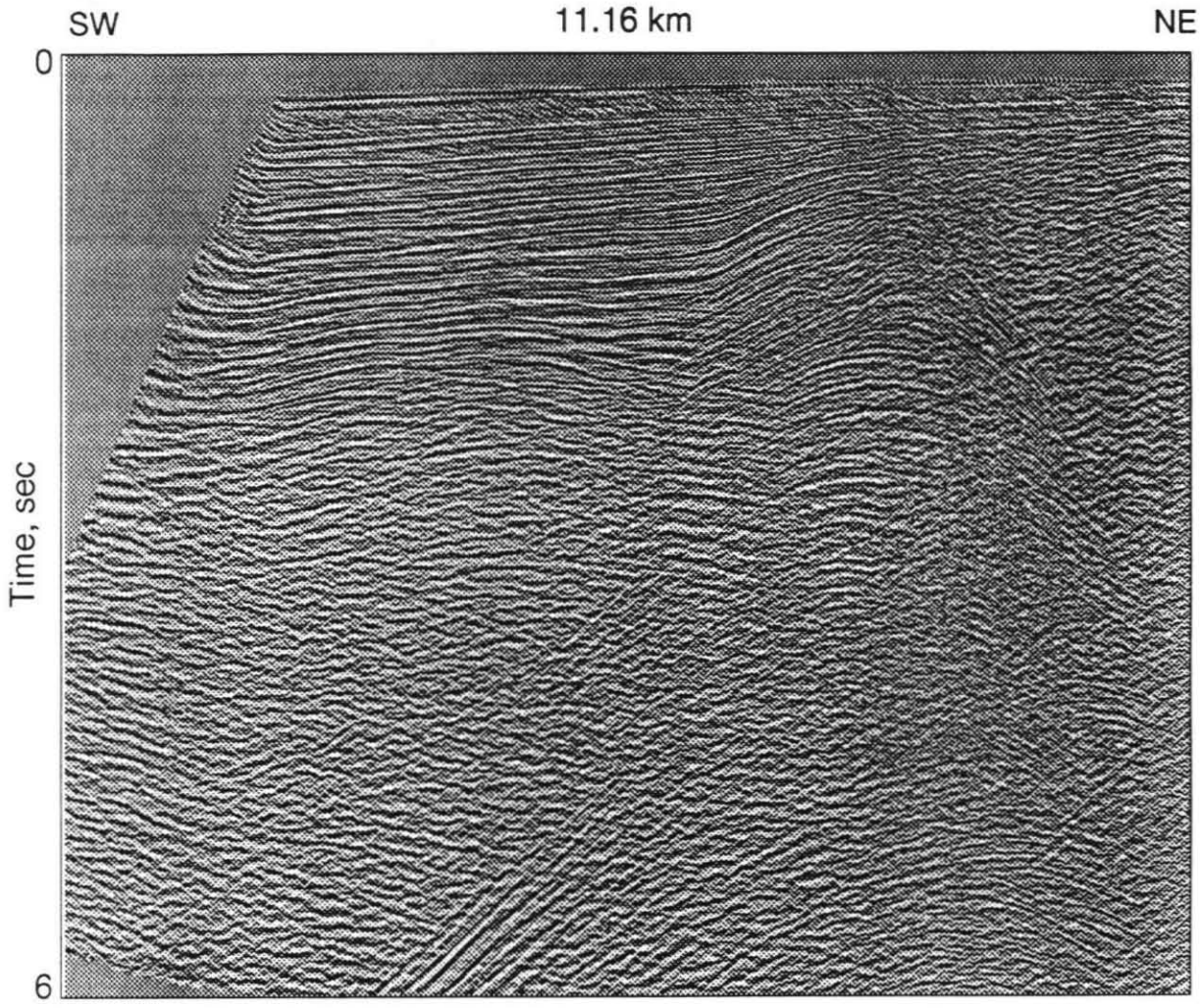

Figure 3.7: A CMP stack of the filtered, multiple-suppressed Hosgri data. AGC has been applied to bring out faint details.

Multiples

Several methods of suppressing multiples exist (e.g., Yilmaz [41], Chapter 8), although none is entirely satisfactory. Most methods rely on the distinction between the apparent velocity of the multiples versus the primaries. These methods tend to use the differential moveout between primaries and multiples for suppressing the multiples, a process that is dependent on the availability of far-offset traces, since the differential moveout on the near-offset traces is minimal. In fact, offset itself can be an effective means of multiple suppression. As was mentioned above, the lack of far-offset traces undercut the effectiveness of the multiple suppression on the eastern end of the survey, and the multiples remain strong in that region. Another problem with the moveout approach is that the velocity distinction between water and the first layers of the sedimentary column is only about 200 to 300 m/s, not enough for reliable differentiation. This problem was exacerbated by the shallow water which resulted in closely spaced, and early-arriving multiples. The result was that the primaries were also largely attenuated by the multiple-suppression process.

A time-domain multiple suppression method in which NMO correction is applied to CMP gathers at the multiple velocity, followed by stacking, subtraction of the stacked trace from the traces in the gather, and inverse NMO, failed to yield acceptable results. An F-K method, in which the CMP gathers are NMO corrected at a velocity between the primary and multiple velocity and then F-K filtered to remove the negative-dipping energy (the multiples) was somewhat more effective, but still removed much of the primary energy in the near-surface reflections (Figure 3.7).

We also applied predictive deconvolution with a variety of parameter selections in

an attempt to suppress multiples and remove the source characteristics. This process had little positive effect and was omitted from later processing.

Another attempt to suppress multiples through the use of near-offset mutes was fairly effective, but degraded other portions of the image to an extent that the technique was also abandoned. A weighted near-offset mute might be more successful, but was not attempted.

Despite our relative lack of success in eliminating the multiples while preserving primaries, the situation is not particularly grave. One often feels more secure working with data in as near an unaltered state as is possible. In this way, one can avoid introducing numerical artifacts and, once the problem areas have been identified, can keep them in mind when interpreting the section. We therefore tended to use the unfiltered data for most of our interpretations, resorting to the filtered multiple-suppressed data only as necessary.

3.3.2 Mutes and Trace Editing

Mutes provide a means of editing undesirable energy from seismic traces in the time-offset domain of the individual gathers. The interactive system allowed us to experiment with the mute settings extensively. We ultimately settled on a simple two-mute strategy for first removing the direct arrival and all prior arrivals, and second, removing the surface wave-train. The surface waves are only a minor problem in the western portion of the survey (Figure 3.8), but in the east, a substantial wave-train developed (Figure 3.9) that required suppression. The surface waves developed due to a slight increase in the near-surface velocity on the east side of the Hosgri fault.

We also interactively edited the data, removing a bad shot, several traces with

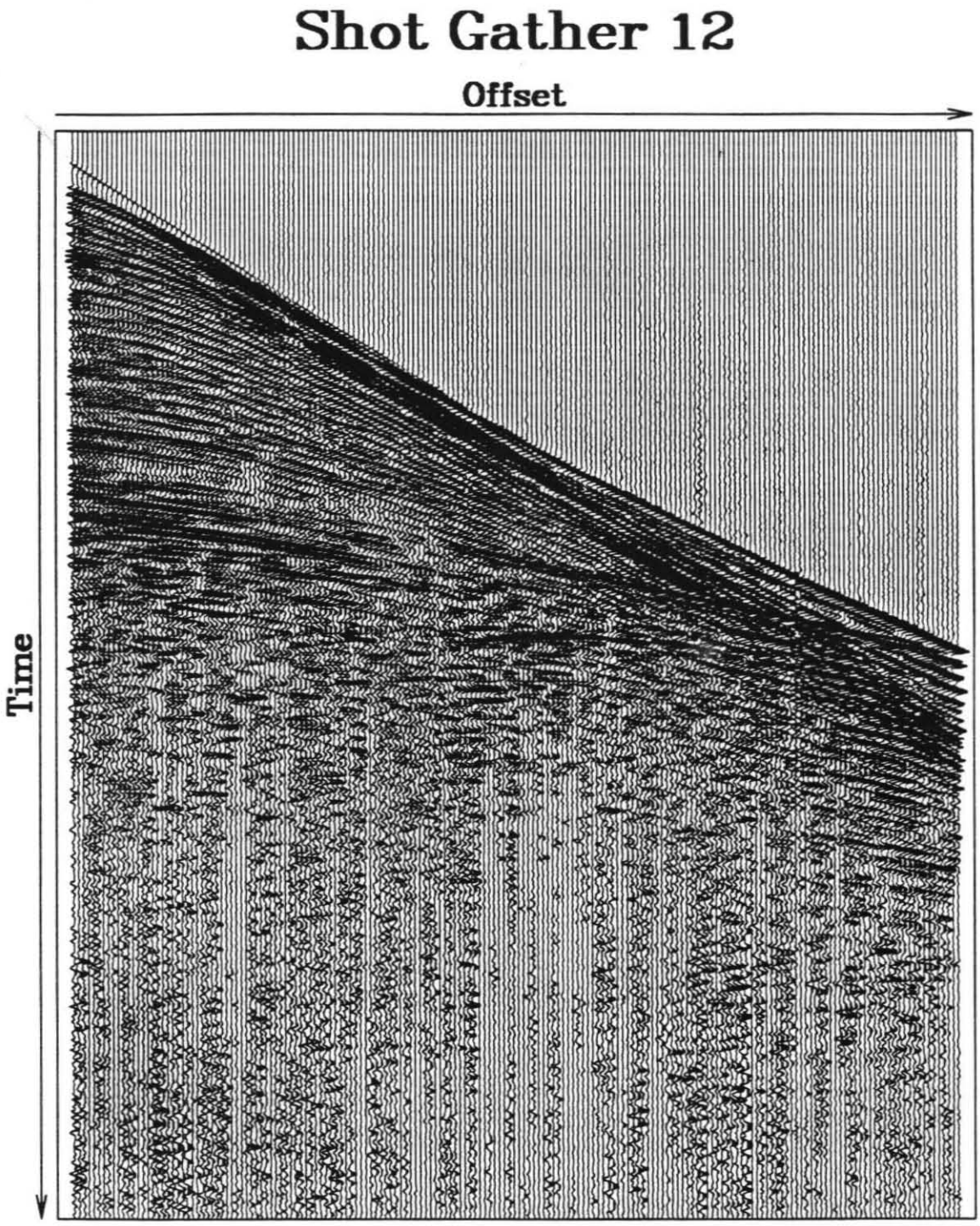

Figure 3.8: A shot gather from the western end of the survey line.

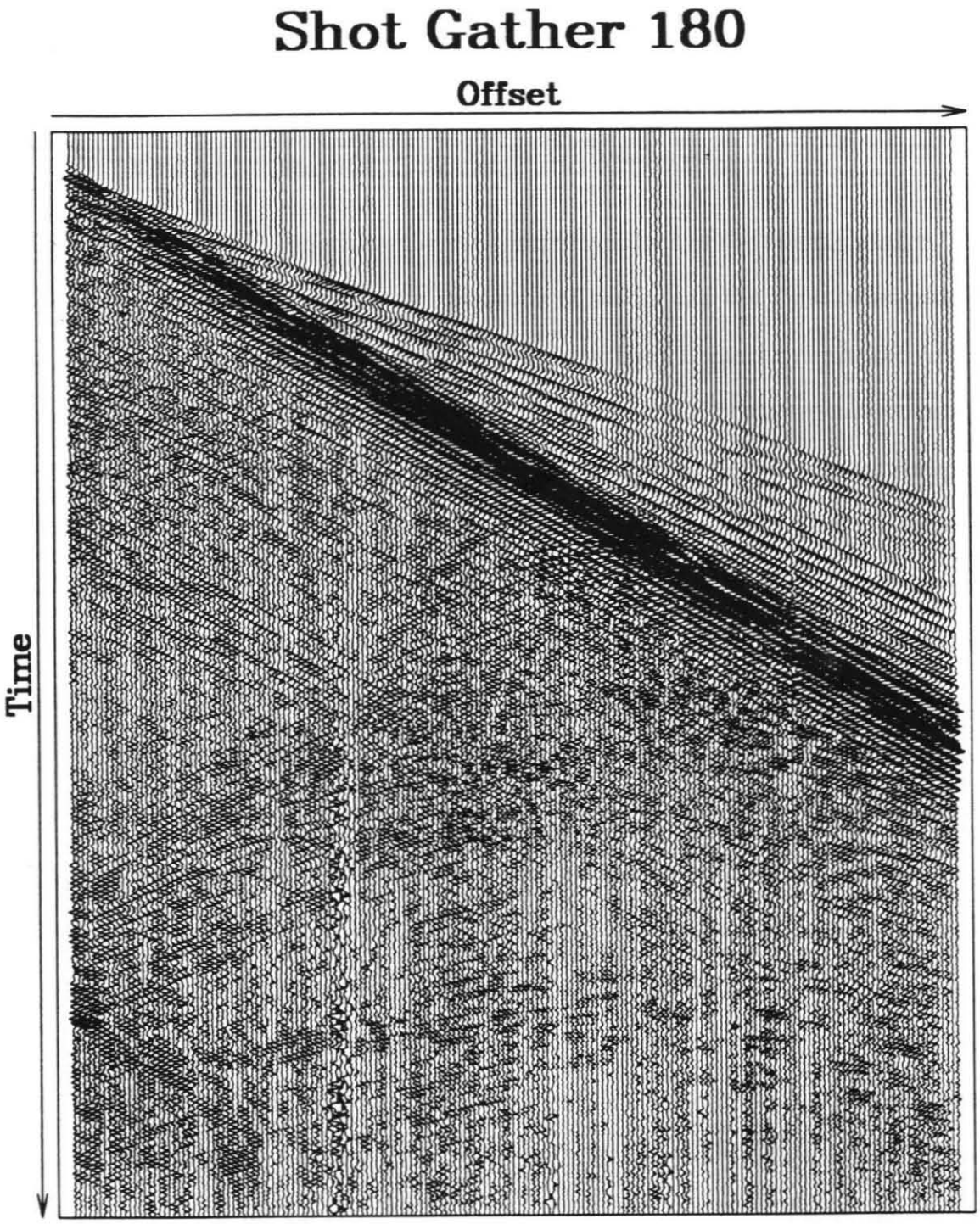

Figure 3.9: A shot gather from the eastern end of the survey line.

large spikes, and one channel on every shot that was either polarity reversed, mistimed, or otherwise malfunctioning.

3.3.3 Velocity Analysis, Stacking, and Migration

The velocity analysis is perhaps the most important task of the seismic analyst. The velocity model directly affects the resolution of stacks and the focus of migrations, as well as the placement of structures by the migration. Correct sub-surface velocity estimation can bring out subtle structures and fine detail, just as incorrect velocity estimates can obscure true structure and even produce deceptive, unreal structure. If our interpretation of a section is to be accurate, we must first image the structure and properly locate the structure within the section.

We performed interactive velocity analysis, using both interactive parameter selection and NMO-corrected midpoint gather movies, as described in Chapter 1. While the performance of the current system does not reach a level sufficient to allow interactive focusing, we were able to analyze a wide variety of velocity models by stacking and migration. A stack of the entire line took approximately 1.5 minutes—a time short enough to allow us to repeat the process literally hundreds of times. We also performed dozens of migrations, both post- and prestack.

The lack of far offsets hampered the velocity analysis in the area to the east of the fault. Further hampering the analysis were a large number of diffractions in the CMP gathers from that area (Figure 3.10, compare to Figure 3.11). These diffractions could not be completely removed by filtering. In addition, the bedding in the eastern segment was weak, irregular, and steeply dipping. Therefore there was little coherent energy to work with once the filtering was performed. These factors

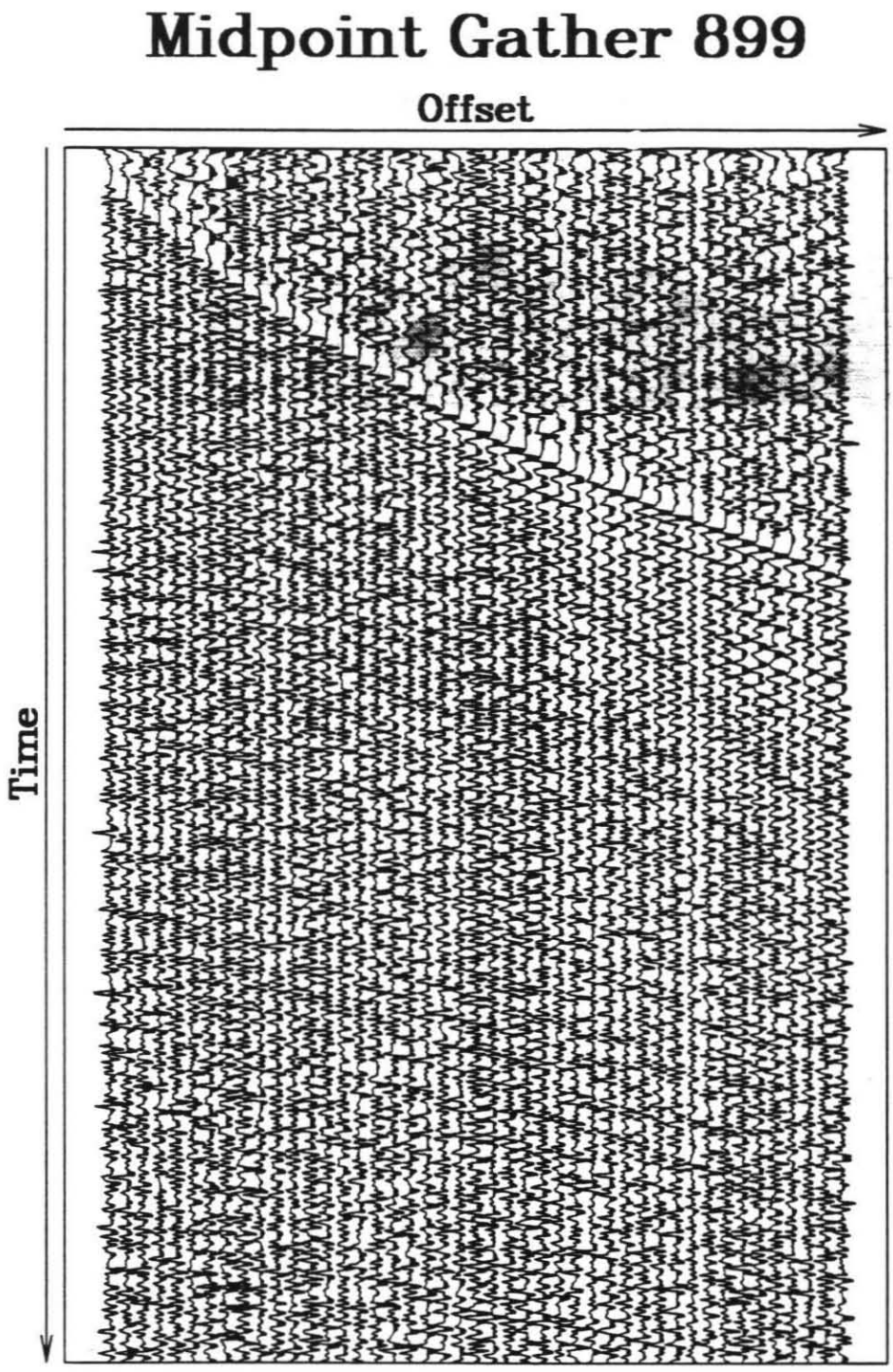

Figure 3.10: A CMP gather from the eastern end of the survey line. AGC has been applied to enhance the later arrivals.

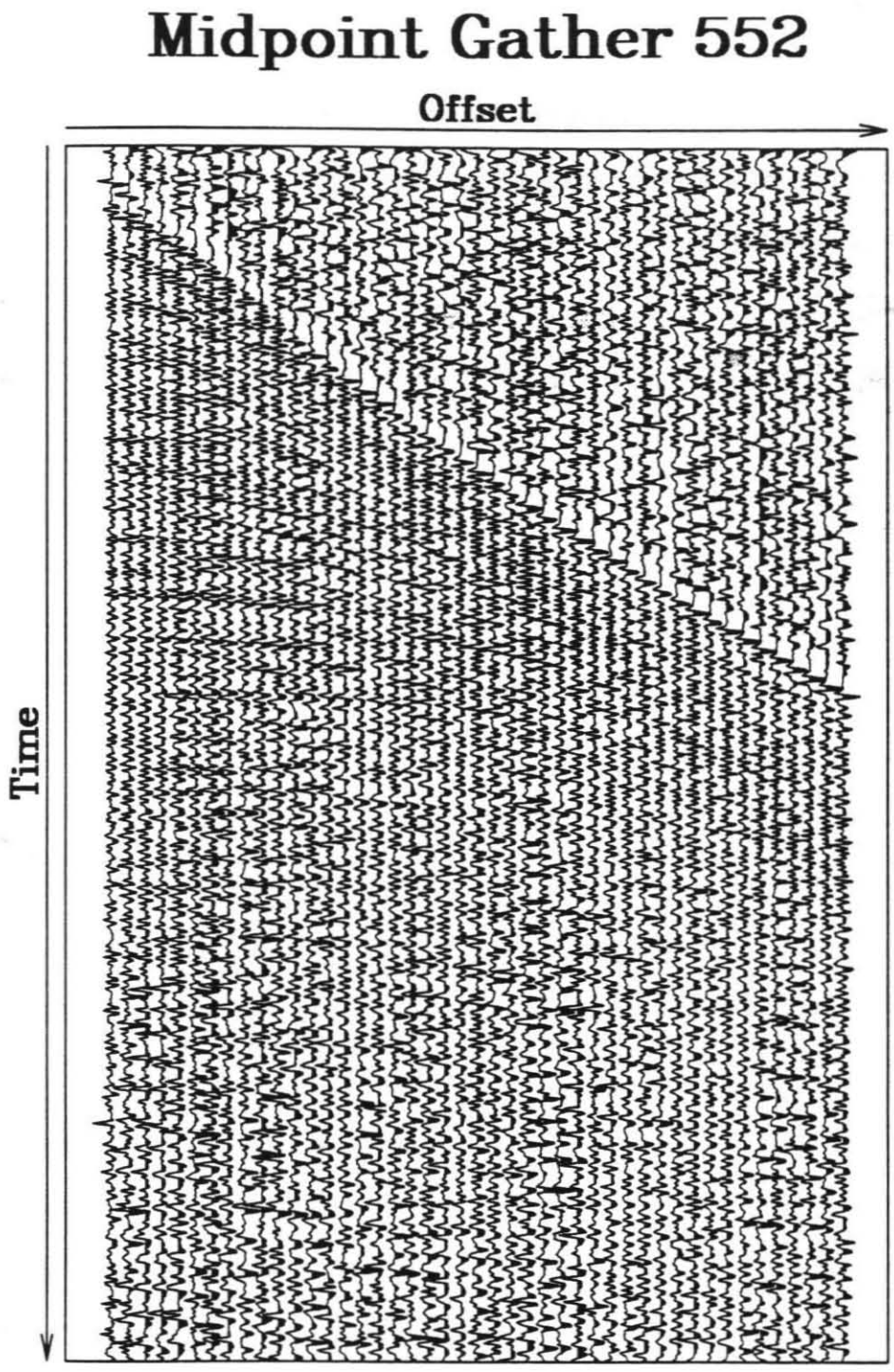

Figure 3.11: A CMP gather from the western end of the survey line. AGC has been applied to enhance the later arrivals.

combined to make the velocity analysis difficult using conventional (NMO) methods. A wide range of velocity models all appeared satisfactory for stacking. We tried models varying from nearly laterally homogeneous to ones with high velocities east of the fault, some had steep gradients, others varied smoothly. The resulting stacks differed only in small details.

While the stacks were stable even when subjected to relatively large variations in the velocity, the migrations, as expected, were quite sensitive to the velocity model. As discussed in Chapter 2, we implemented three methods of migration. The simplest, a post-stack Kirchhoff time migration allowed for minor lateral velocity variations, but not to the extent required by some of the models we tested. In those cases, the time migration created severe artifacts that prevented proper interpretation. The other post-stack migration allowed for major lateral velocity variations, and was fairly effective but, because it was post-stack, could not properly handle the steep dips encountered in the section (this problem has more to do with the assumptions of stacking than with a failure of the migration). The prestack Kirchhoff migration was needed to handle the lateral velocity variations and dipping structures that we encountered.

Because of the substantial computing power of our imaging system, the migrations could be done relatively quickly (~5–10 minutes for post-stack migration), allowing us to do a large number of iterations with various velocity models. By repeatedly migrating with different velocities, we were able to hone in on the most promising models, and then perform the slower, more accurate prestack migration. Here, interactive migration, if available, would have greatly simplified and speeded

the process.

Surprisingly, the models with high velocities to the east of the fault were seriously over-migrated. Those with lower velocities at depth and in the eastern area provided the best images. Other studies (e.g., [26,23,10]) have used considerably higher velocities in the area to the east of the Hosgri. These studies may have been led astray, as we were, by the NMO analysis. The basement rock to the east of the survey may display low seismic velocities due to fracturing, anisotropic effects, or both.

The initial high-velocity models we used in the migrations showed classic over-migration artifacts: heavy, upturned, semicircular arcs cutting through bedding (Figure 3.12). As those artifacts disappeared with the development of slower models, we detected more subtle artifacts such as minor arcs and mis-tied diffractions (Figure 3.14). Continued lowering of the velocities improved the continuity of near-fault structures and bedding (Figure 3.16). Further adjustments to the velocity model yielded better detail in parts of the image (Figure 3.18) and undermigrated areas in others (Figure 3.20). We made no attempt to develop a single ideal velocity model, however, because of the enormous time needed to develop an internally consistent model. We chose instead to develop a model of sufficient accuracy to give a good overall image, while noting the details of other images. Again, a fully interactive system would have been a great advantage throughout this process.

In Chapter 1 we discussed the advantages of an interactive system for the interpreter. During the migration velocity analysis, discussed above, we were able to get a strong feel for the robustness of the imaged structures. The geological interpretations we make here are the result of studying many images, not just the ones displayed.

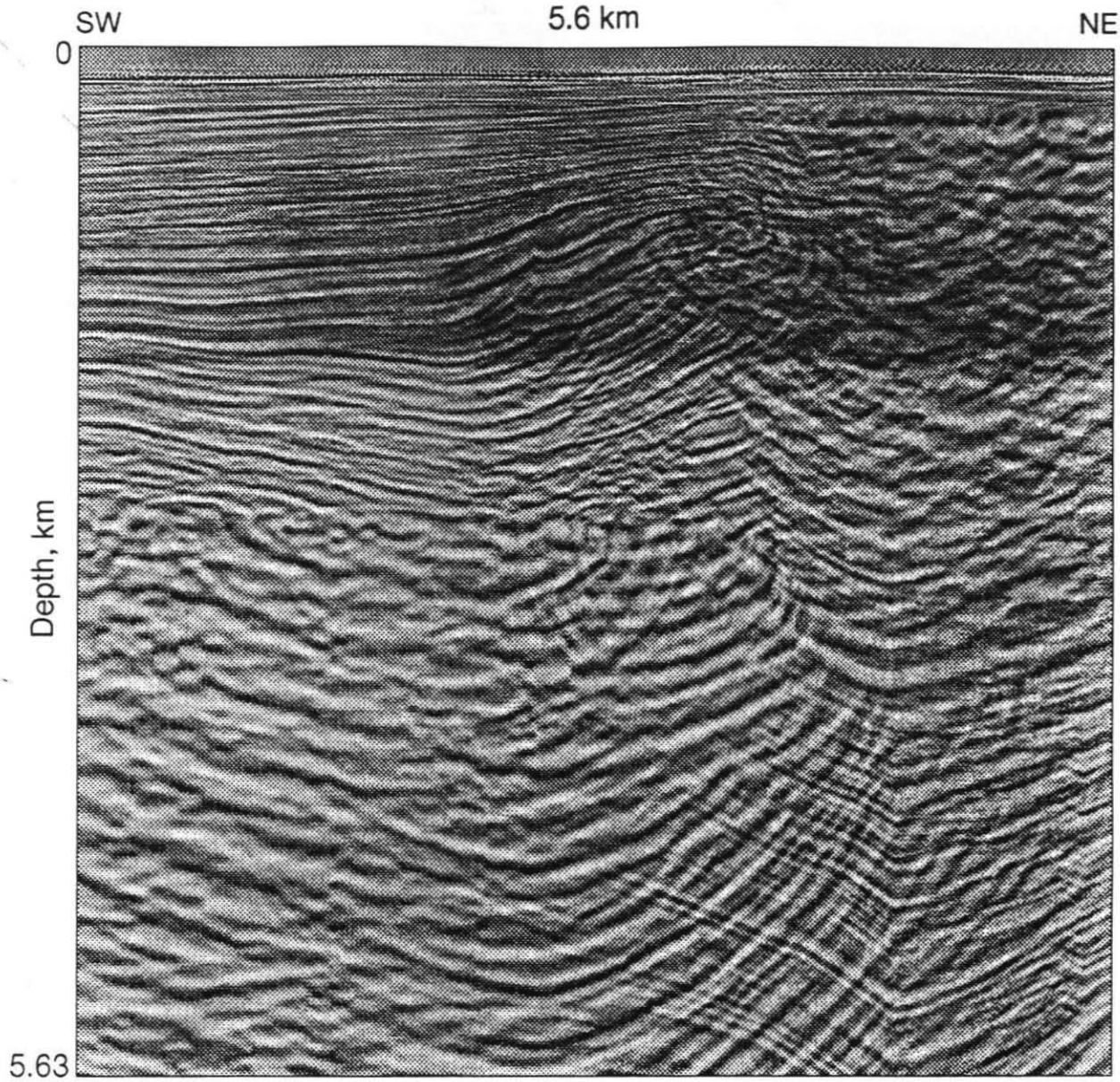

Figure 3.12: An example of a grossly over-migrated image from a post-stack migration. AGC has been applied to bring out the faint details. The velocity model for this migration is shown in Figure 3.13.

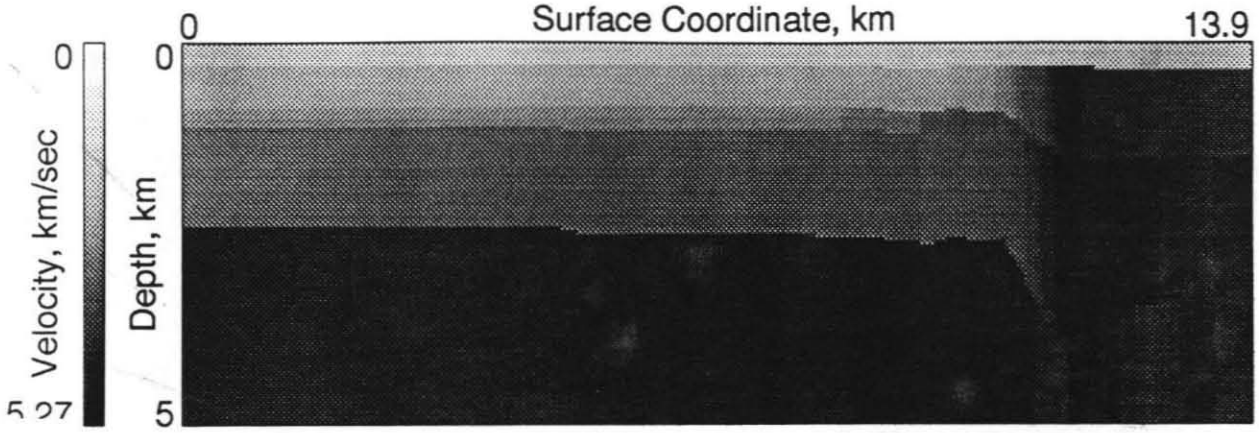

Figure 3.13: The velocity model used by the migration shown in Figure 3.12.

Some of the features are subtle, and even arguable, but we are more confident of our interpretation because we have imaged them over a wide range of velocity models.

3.4 Conclusions: Geological

The most important feature in the seismic profile is the several-stranded, nearly-vertical fault running from the near-surface to about 3 sec on the stacked section (Figures 3.22 and 3.5). The fault can be detected by the numerous truncated beds, changes in slope of beds, diffractions, and by a general change in the character of the reflection energy from one side to the other. Figures 3.6, 3.23, 3.24, and 3.25 show more detail of the stack. The prestack migrations show detail and depth-corrected structure that is not available in the stacks (Figures 3.18, 3.26, 3.27, 3.28). It is not clear whether the imaged fault breaks the surface, since there is no apparent fault scarp on the seafloor. The bathymetry indicates that the seafloor depth is relatively constant along strike, we would therefore not expect to find a significant scarp if

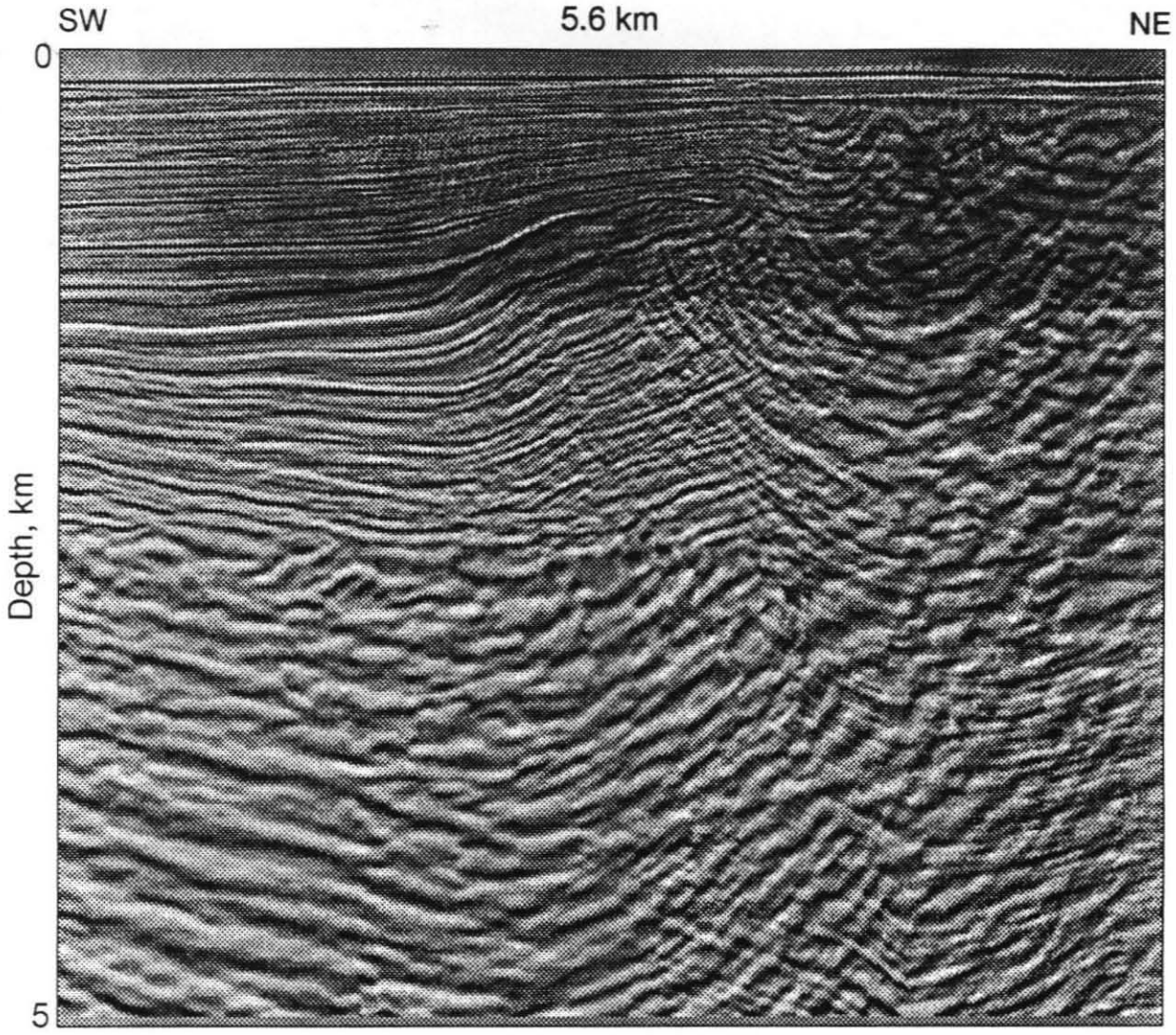

Figure 3.14: An example of a mildly over-migrated image from a prestack migration. AGC has been applied to bring out the faint details. The velocity model for this migration is shown in Figure 3.15.

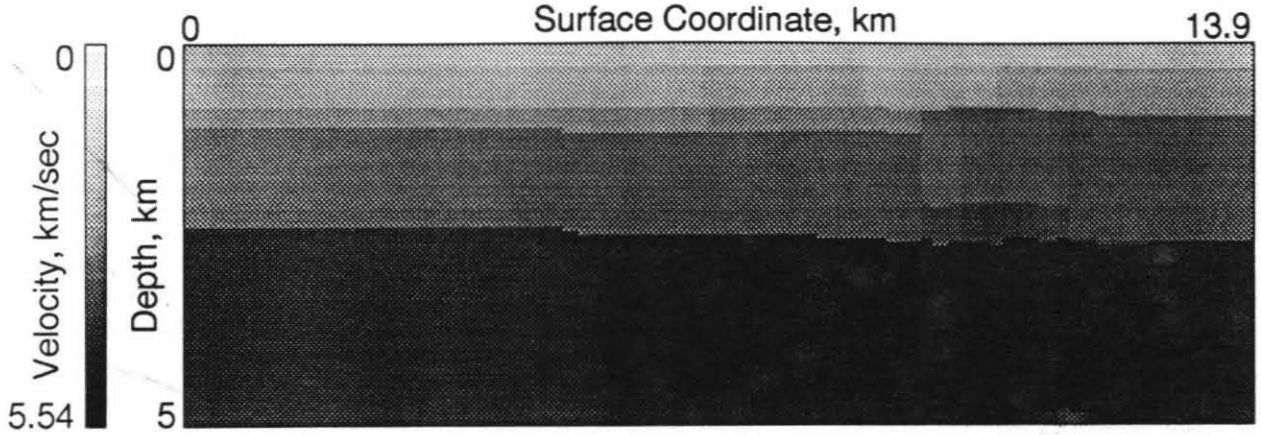

Figure 3.15: The velocity model used by the migration shown in Figure 3.14.

the fault were strike-slip. The reflector just below the seafloor is truncated, however (Figure 3.24). We can therefore infer that there has been relatively recent activity on the fault. Based on the dip and the lack of a scarp, the fault is very likely strike-slip. The amount of slip on the fault cannot be estimated, but is probably not large since some of the upper beds appear to be nearly continuous, only changing dip at the fault.

There is little evidence, on the other hand, for any recent thrust or reverse faulting. Working with the same section of the same line, McIntosh et al. [26] indicate the antiform just west of the fault trace and state "Here, all stratigraphic units, including the most recent, thin onto the structure and are uplifted and tilted as well." A close inspection of Figures 3.6 and 3.18 shows that the beds lapping onto the deformed beds appear undisturbed. The overlying beds are also undisturbed, as is the sea floor. While compressional deformation was obviously part of the formation of the basin, it does not appear that the major folds seen in our images are active.

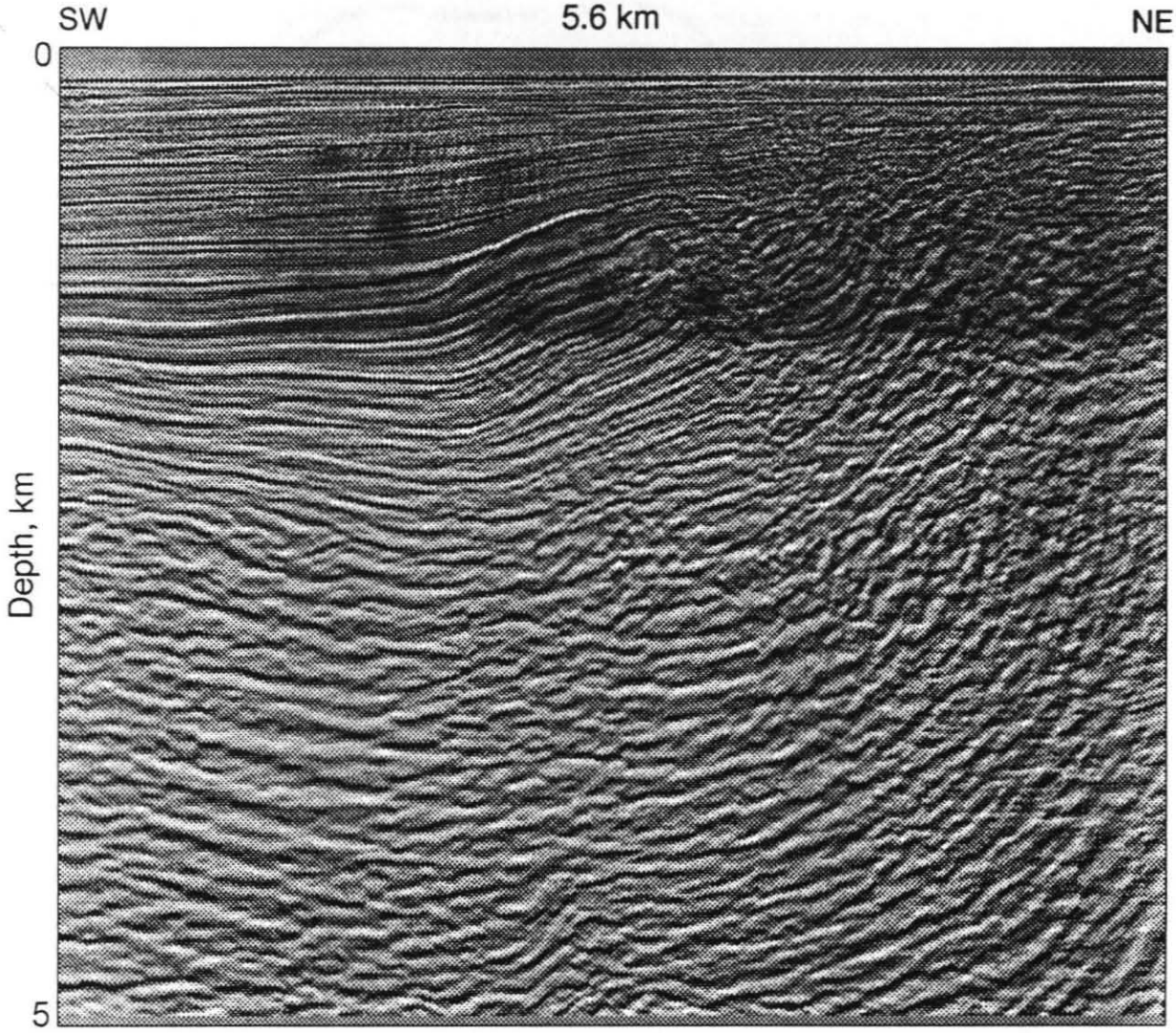

Figure 3.16: An example of a very slightly over-migrated image from a prestack migration (note the slight upturned arcs on the dipping beds in the upper center of the image). AGC has been applied to bring out the faint details. The velocity model for this migration is shown in Figure 3.17.

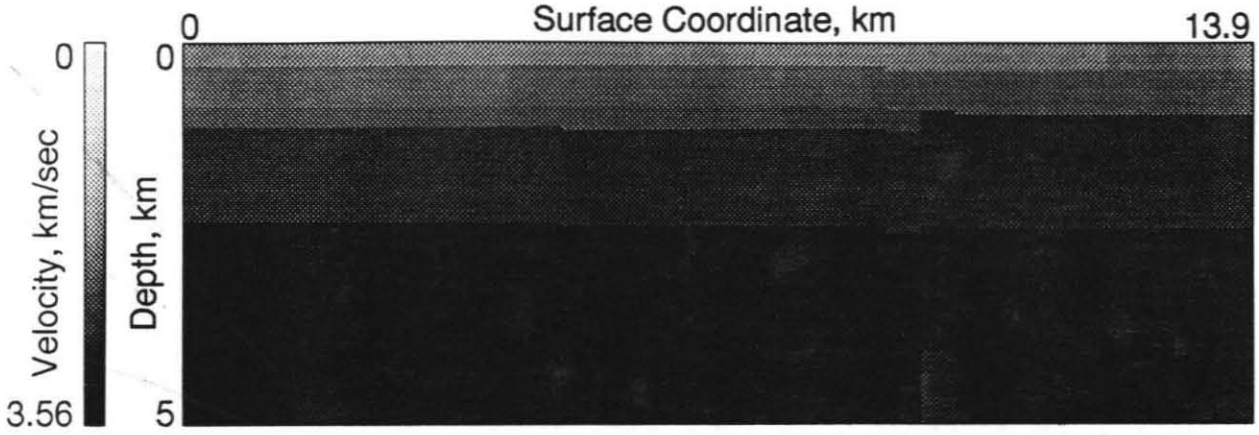

Figure 3.17: The velocity model used by the migration shown in Figure 3.16.

There does appear, however, to be evidence for a thrust fault in the section (Figures 3.26 and 3.18) as well as the vertical fault. The thrust fault we observe is oriented in much the same way that Louie [23] describes it. It appears that the vertical fault truncates the thrust fault, however. It may therefore be a fossil thrust, or one with only episodic or intermittent deformation. The undeformed overlying beds seem to confirm this belief.

A series of short, dipping, truncated beds just to the east of the fault (Figure 3.26) have much the same dip and character as those just below the thrust described above. It is possible that they are beds from the same structure, with the higher beds being brought into the image (in a three-dimensional sense) by the strike-slip fault. If the truncation was caused by thrust faulting, the orientation of the fault plane would tend to indicate generally east-west compression, rather than the northeast-southwest compression that is believed to dominate the area.

The strong, steeply-dipping energy labeled "B" in Figure 3.5 is not the result of

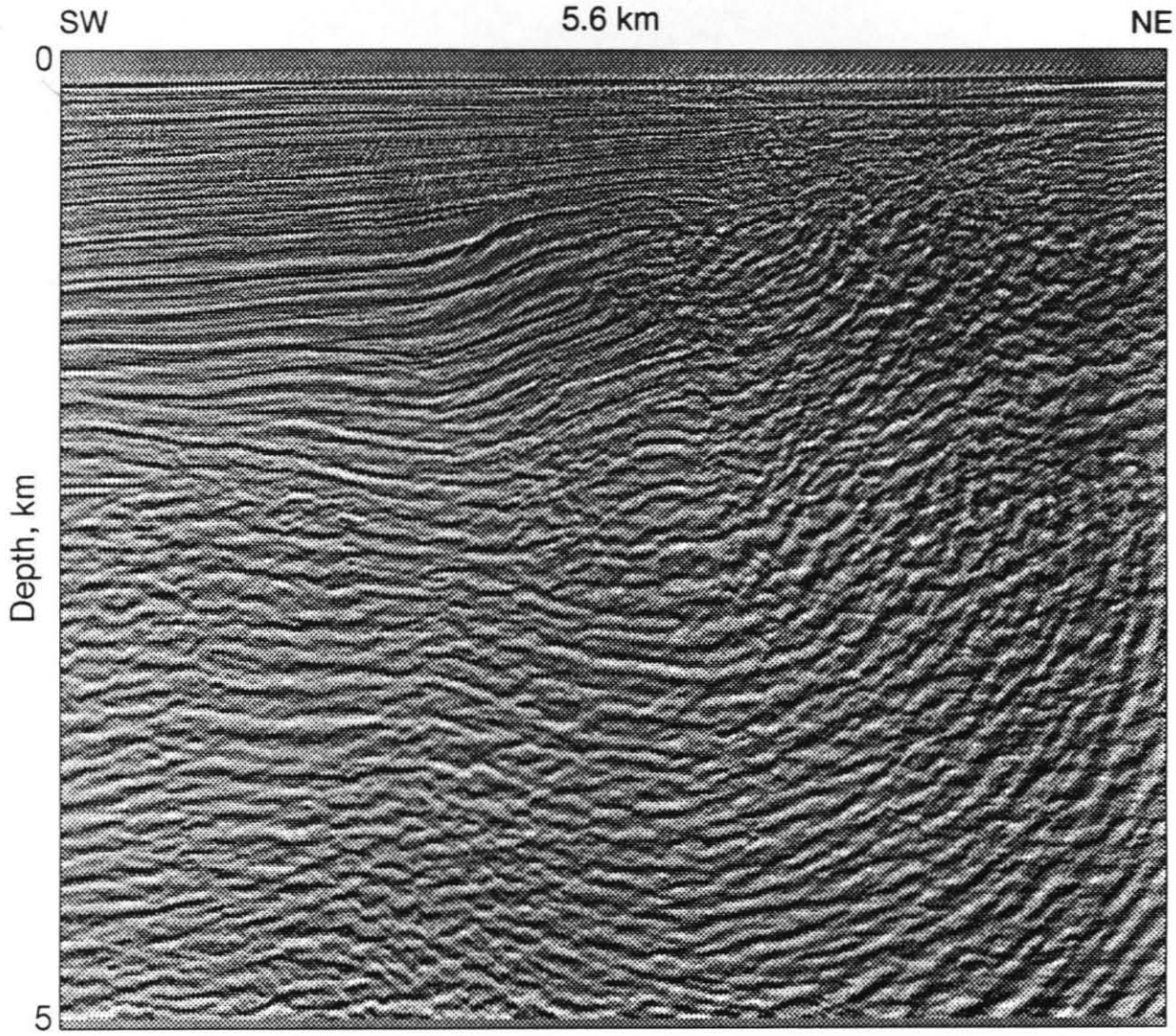

Figure 3.18: An example of a well migrated image from a prestack migration. Note that the artifacts on the right edge and at depth are the result of edge effects of the migration. AGC has been applied to bring out the faint details. The velocity model for this migration is shown in Figure 3.19.

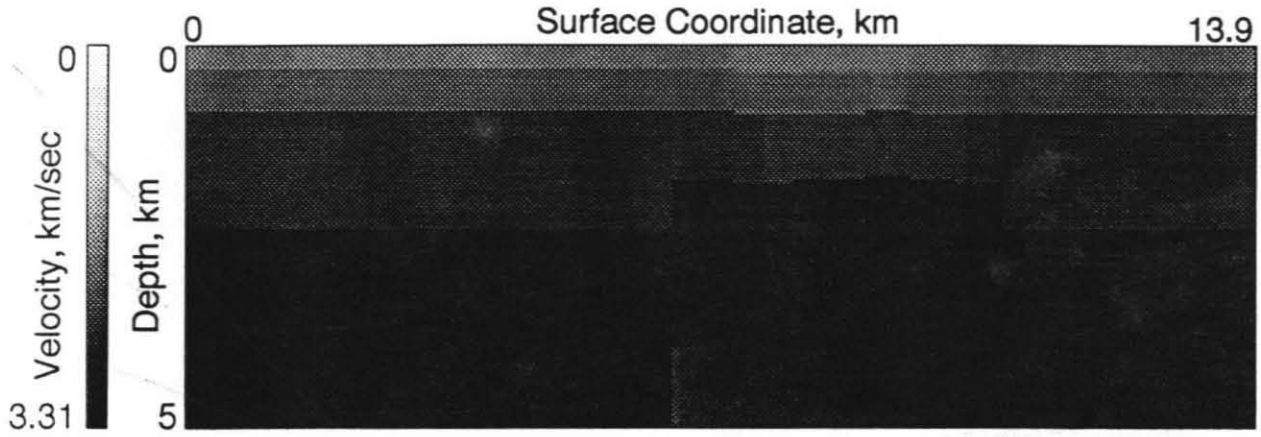

Figure 3.19: The velocity model used by the migration shown in Figure 3.18.

geologic beds in that position. It is, rather, a reflection traveling at approximately the water velocity in a nearly-horizontal direction from a fault or other three-dimensional structure.

In Figures 3.6 and 3.18 we can clearly see that the first sub-seafloor reflector east of the fault is extremely rough and uneven, while the seafloor to the west is flat and even. It is very likely that the three-dimensional scattering from this rough reflector is the cause of the large number of low-velocity arrivals in the eastern midpoints (Figure 3.10), and the linear coherent noise discussed above. It is also very likely that this reflector caused the general degradation of the data quality east of the fault. The cause of the roughness is not known. It may be the result of fracturing of the rock related to the formation of the adjacent San Luis anticline, which would also explain the lower-than-expected seismic velocities. Another possibility is that the reflector was at one time above sea level, and the roughness is the result of erosion.

It appears, then, that the Hosgri fault in the area of EDGE line RU-3 is a near-

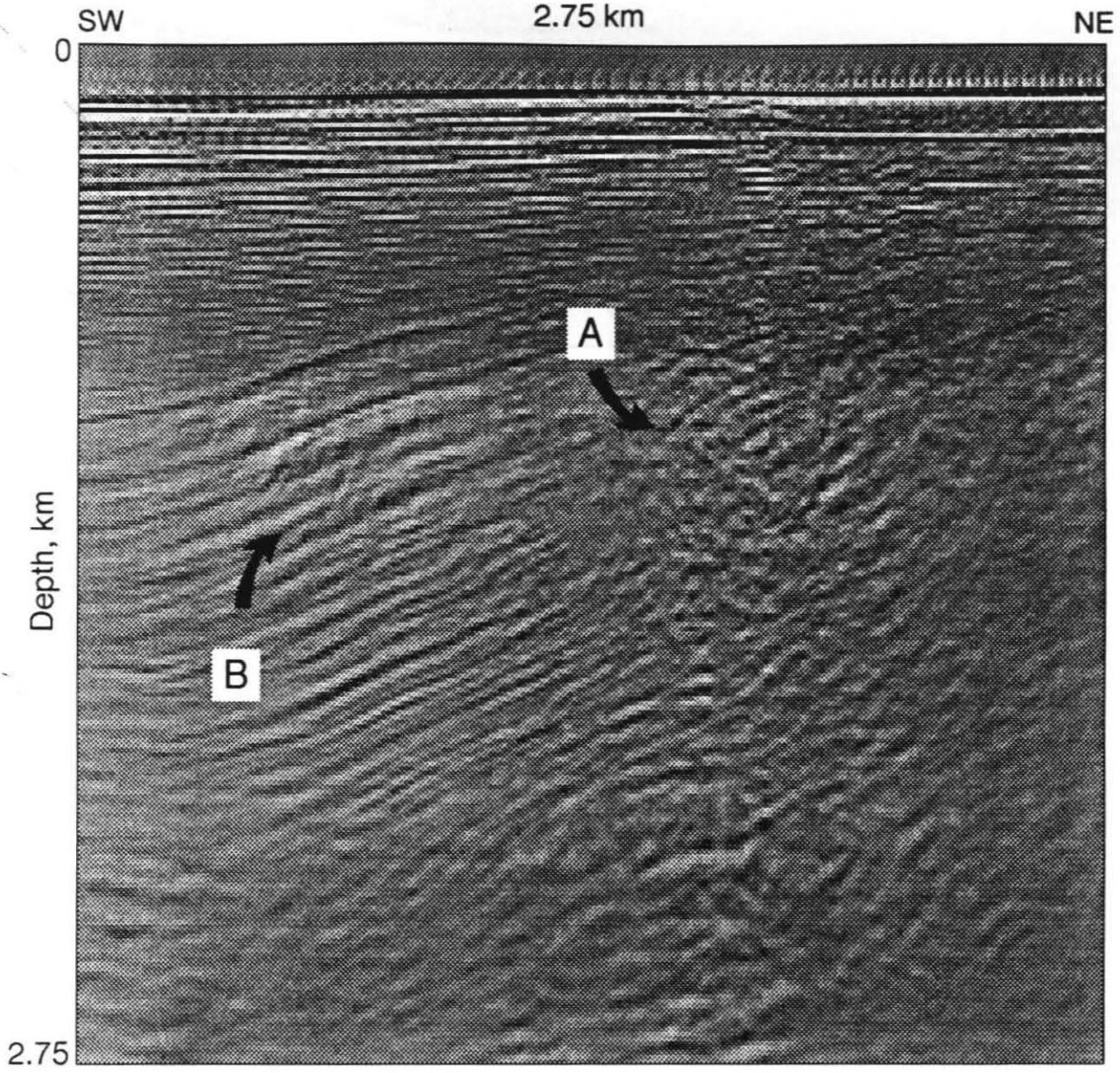

Figure 3.20: An example of a slightly under-migrated image from a prestack migration. This image is shown at twice the magnification of the earlier migrations. No AGC was applied in order to emphasize the uncollapsed diffractions (A) and the break-up of the dipping beds (B). The velocity model for this migration is shown in Figure 3.21.

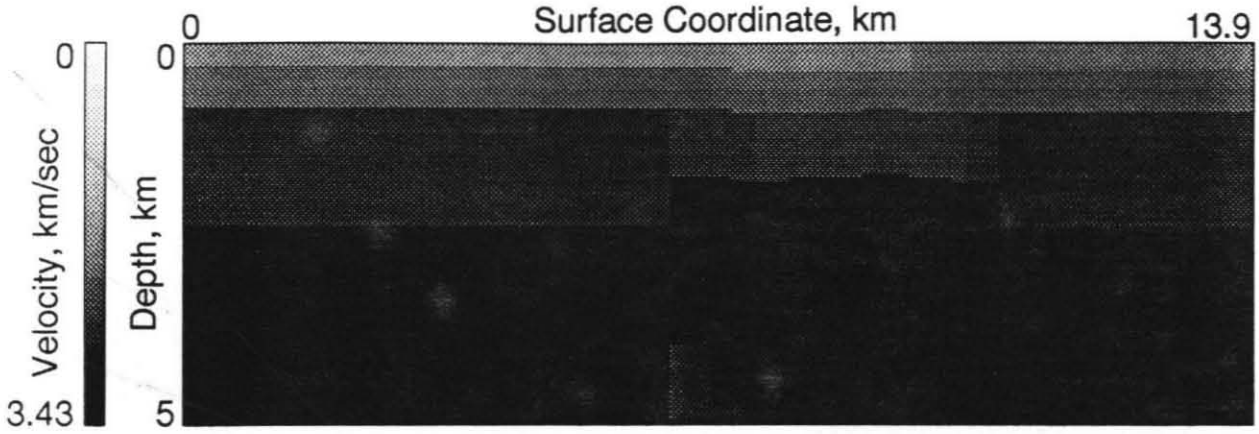

Figure 3.21: The velocity model used by the migration shown in Figure 3.20.

vertical fault that shows recent, if not current, deformation. The style of deformation is probably strike-slip. There are also structures that indicate significant compressional stress and thrust or reverse faulting, but these do not appear to have been active for some time.

3.5 Conclusions: Processing

One of the goals of working with the Hosgri data was to test and evaluate the interactive imaging system. More important than the actual performance of the hardware and software, which was adequate, was the evaluation of the concepts presented in Chapter 1. Explicitly stated: Does interactive imaging provide significant advantages over conventional methods? Our work on the Hosgri and other data sets has convinced us that the answer is unquestionably "yes." Throughout the processing sequence we were impressed by the advantages of the interactive system, and were made acutely aware of the drawbacks of conventional methods of processing when

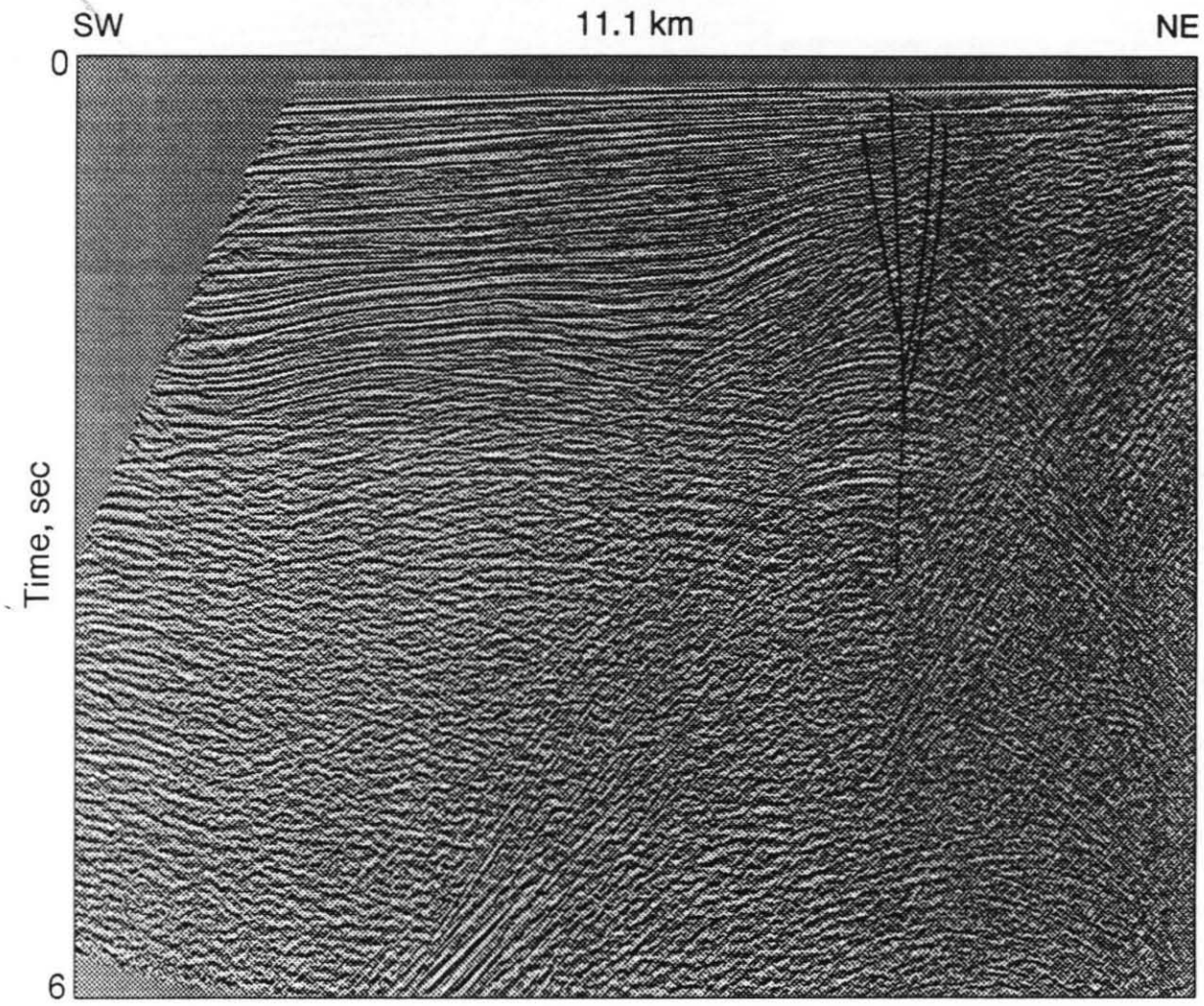

Figure 3.22: An interpreted CMP stack of the raw Hosgri data. Figure 3.5 is an uninterpreted version of this figure. AGC has been applied to bring out faint details.

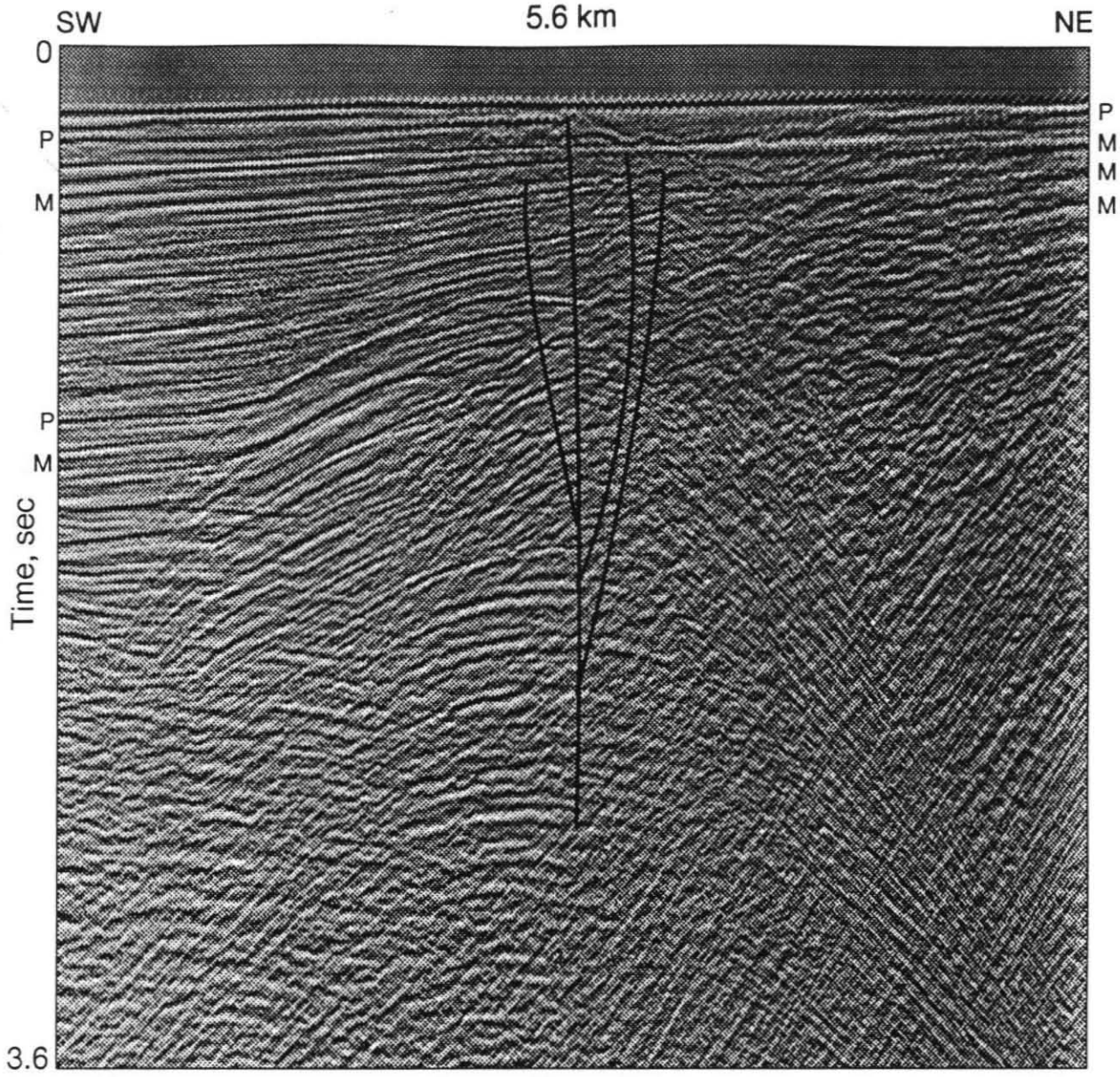

Figure 3.23: Detail of the Hosgri CMP stack (Figure 3.5) with interpretation. This image is enlarged by a factor of two over Figure 3.5. Figure 3.6 is an uninterpreted version of this figure. AGC has been applied to bring out faint details.

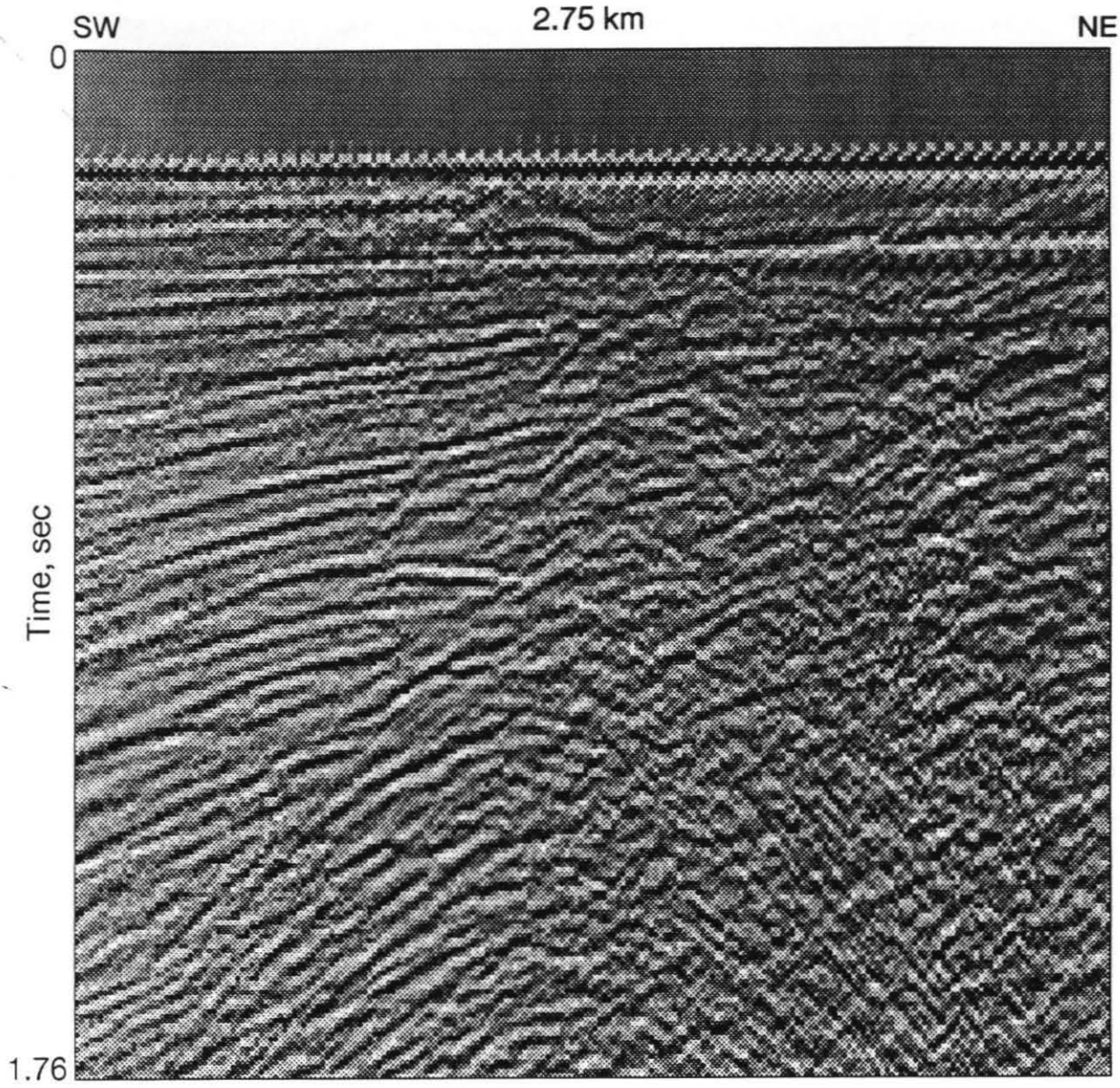

Figure 3.24: Detail of the Hosgri CMP stack (Figure 3.5). This image is enlarged by a factor of four over Figure 3.5. AGC has been applied to bring out faint details.

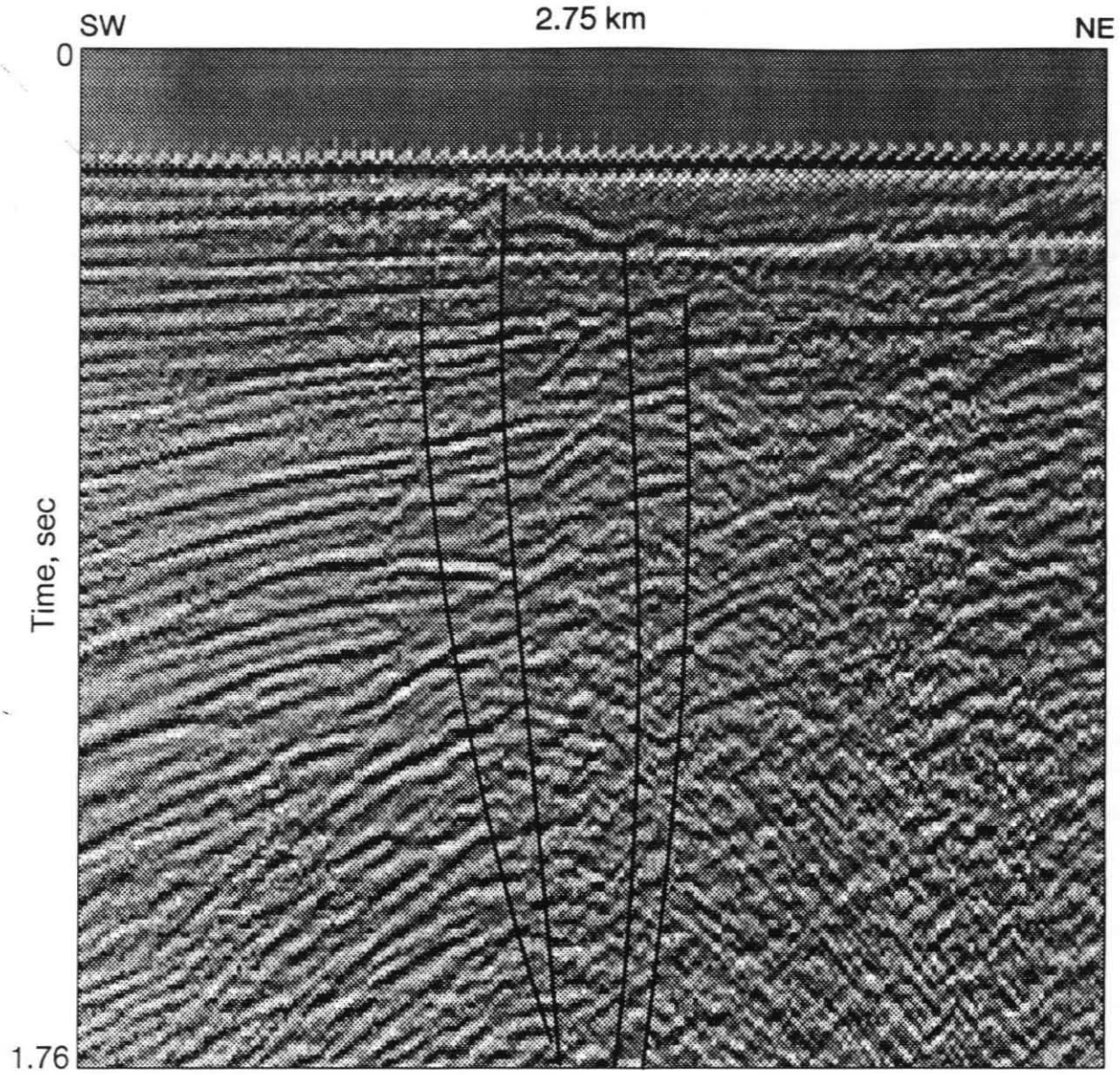

Figure 3.25: Detail of the Hosgri CMP stack with interpretation. This image is enlarged by a factor of four over Figure 3.5. Figure 3.24 is an uninterpreted version of this figure. AGC has been applied to bring out faint details.

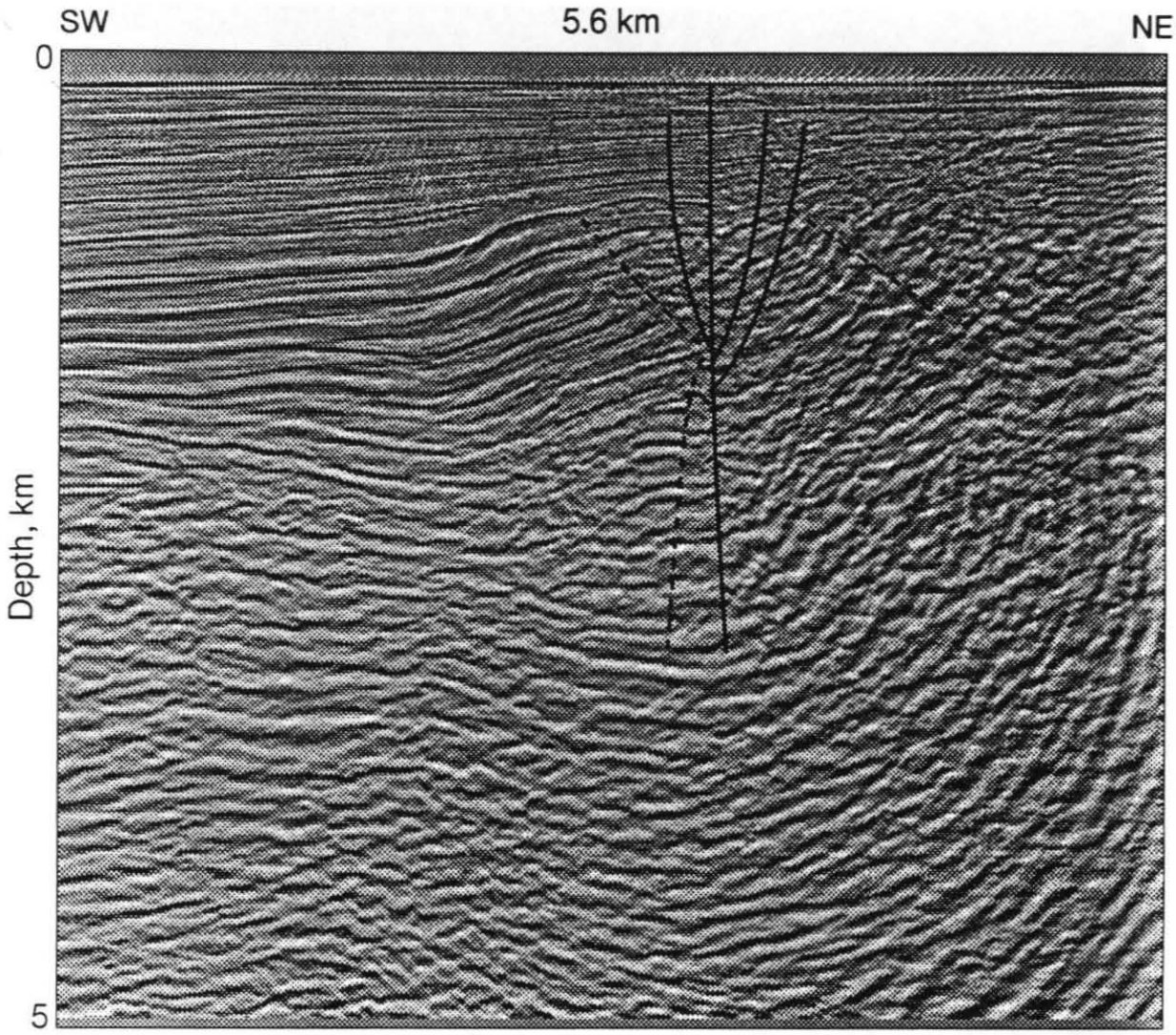

Figure 3.26: An example of a well migrated image from a prestack migration, with interpretation added. Figure 3.18 is an uninterpreted version of this figure. AGC has been applied to bring out the faint details. The velocity model for this migration is shown in Figure 3.19.

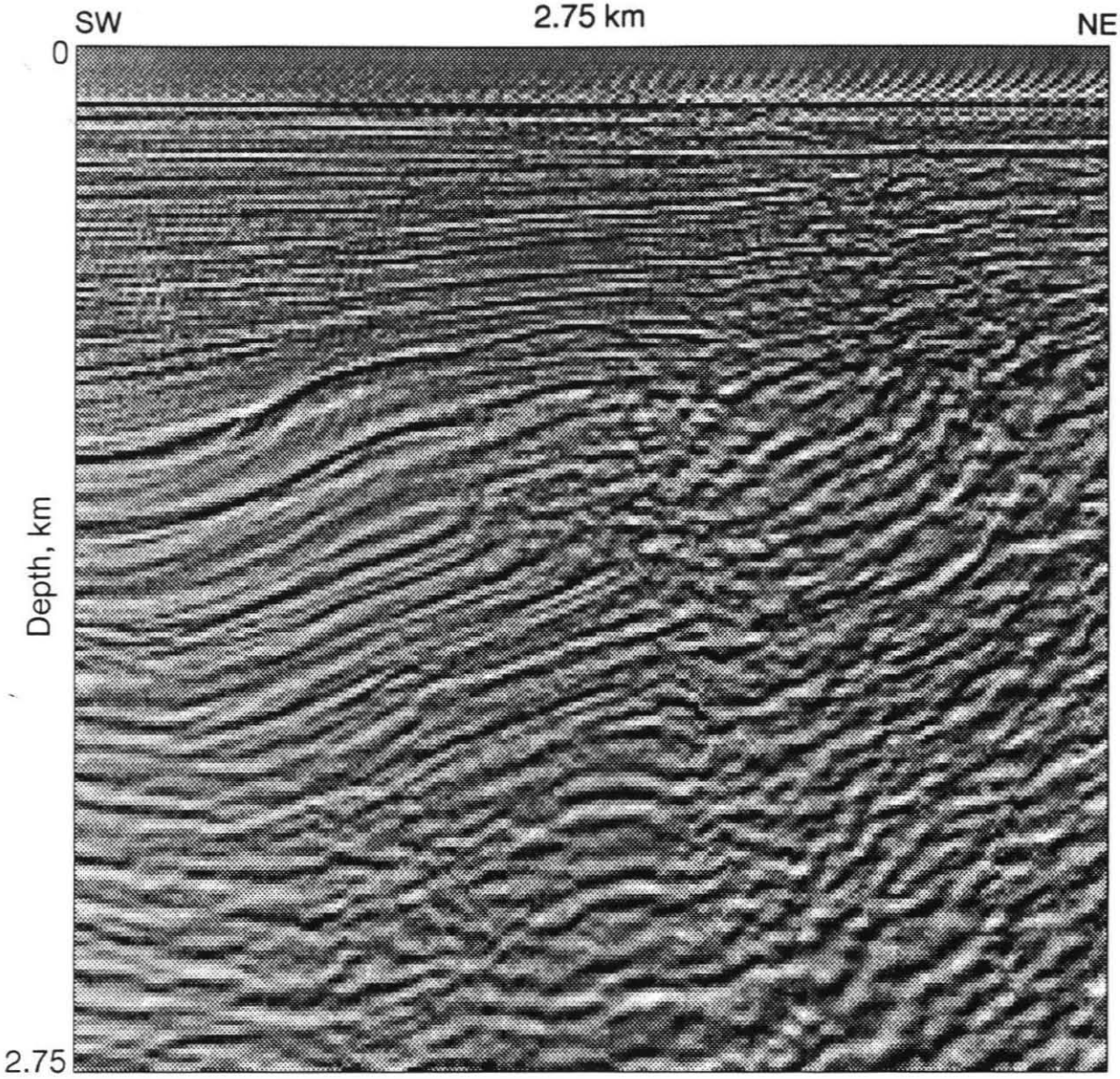

Figure 3.27: Detail from Figure 3.18. This image is enlarged by a factor of two over Figure 3.18. AGC has been applied to bring out the faint details. The velocity model for this migration is shown in Figure 3.19.

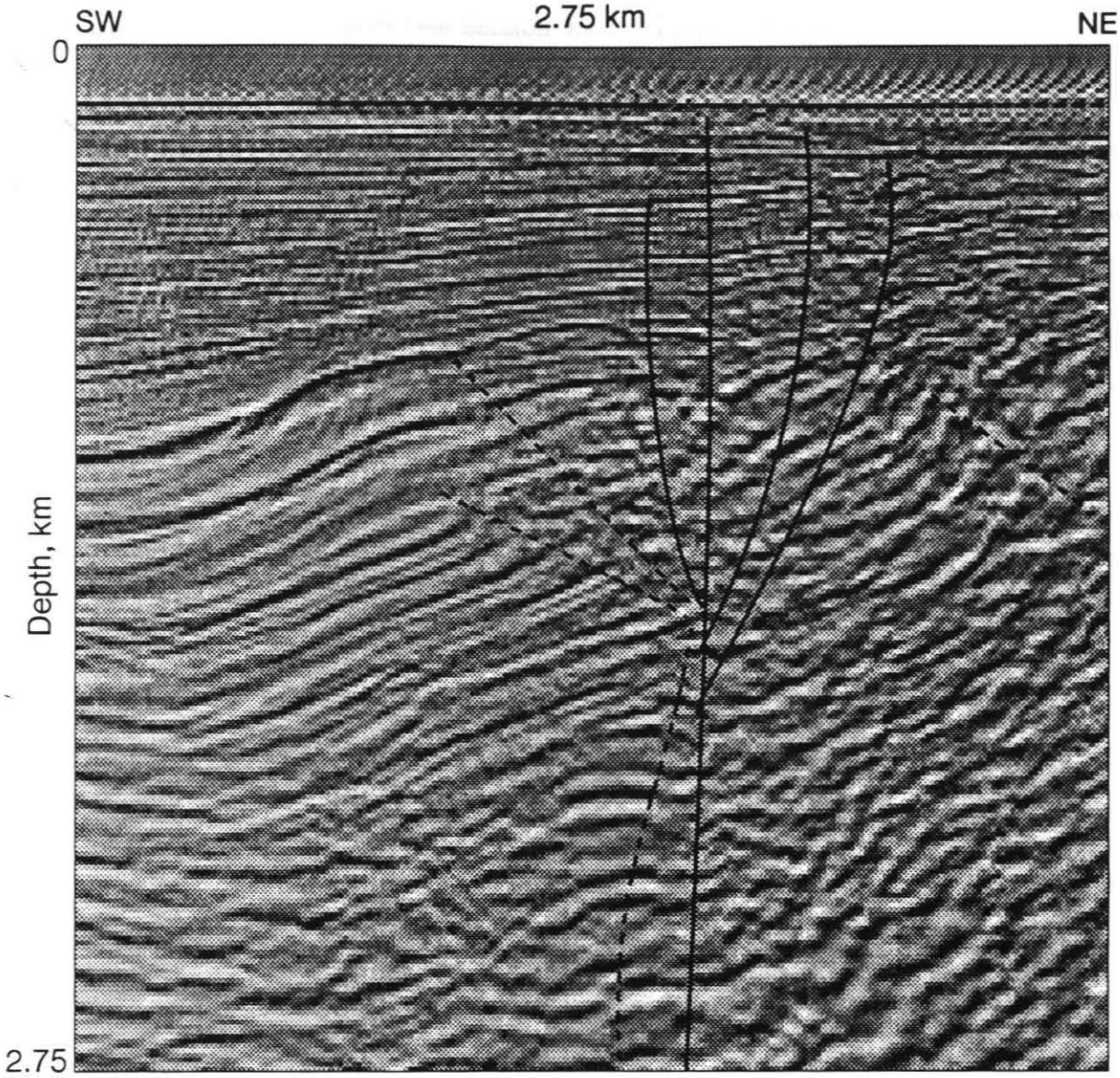

Figure 3.28: Detail from Figure 3.18, with interpretation added. Figure 3.27 is an uninterpreted version of this figure. AGC has been applied to bring out the faint details. The velocity model for this migration is shown in Figure 3.19.

we had to resort to them (see Section 3.3.1: "Preprocessing").

One of the most noticeable difficulties in conventional processing schemes is the difficulty of moving backwards in the processing sequence. As each processing step is completed, intermediate files are produced and those files become the input to the next process. If later processing reveals difficulties with earlier processing, and it almost always does, the sequence must be repeated from the point that the processing changes. While this is true of interactive imaging as well, the philosophy of performing processing tasks "on the fly" rather than through a series of intermediate steps, allows for much easier processing. For instance, if after stacking the data we notice that the section is contaminated by ground roll, we can simply alter the mutes and restack the data, regardless of the intermediate processes, which would be redone on every stack in any case.

This problem was most obvious as we attempted several F-K filters and multiple suppression methods. Upon stacking the data we were inspired to alter the filter settings, which required us to redo not only the F-K filtering, but also the multiple suppression stage. In any case where we altered the F-K filter, multiple suppression, or deconvolution parameters, we were required to reload the data into the interactive system. The advantages of having those processes built into modules within ISIS are plain: we would be able to alter the parameters of any of the processes and directly see the results not only in the data gathers, but in the stacked or migrated sections. Even if the current technology prevented these processes from being fully interactive, the time savings and improvements to the images would be substantial. In addition, under the conventional processing scheme it is almost impossible to judge

the interplay and tradeoffs among the parameter selections for different processes. The interactive system, however, allows several sets of parameters to be active at once, giving the analyst the ability to examine these relationships.

While we did not have the performance necessary to interactively focus the image, we did get a glimpse of the process and its promise. As we discussed above, the NMO velocity analysis seriously misguided us as to the velocity of the region to the east of the Hosgri fault. By repeatedly migrating the image we were better able to focus some parts of it. Each additional increment of performance would allow us to better fine-tune the model and focus the image. Ultimately, interactive migration would produce a nearly-optimum focus, as well as confidence in the analyst that the parameter space had been fully explored.

Again, while the performance of the system did not give us the "feel" of the image (i.e., an intuitive sense of the way it responded to changes in the velocity) we were able to study images produced by a wide range of velocity models, and to observe how features we were trying to interpret responded. In this way we were able to make our interpretations with more confidence than we otherwise could have. Ultimately, when we can feed the interpretation back into the system as an input model, we will also have a direct means of testing our models.

The success of this prototype interactive system has convinced us to proceed with further developments. Having proved the concept, we intend to add more processing modules, as discussed in Chapter 2, and to investigate means of improving the performance. There remain no significant technical barriers to prevent the fully interactive system we have discussed. The software foundation is in place, and several

hardware platforms exist to meet the computational and I/O demands of the system. The only barrier to full implementation is cost, and as the price/performance ratio of computers continues to fall, interactive seismic imaging will be increasingly affordable.

Bibliography

[1] T. Atwater. Implications of plate tectonics for the Cenozoic tectonic evolution of western North America. *Geol. Soc. of Am. Bull*, 81:3513–3536, 1970.

[2] D. H. Clark, N. T. Hall, and D. H. Hamilton. Structural analysis of late Neogene deformation in the central offshore Santa Maria basin, California. *J. Geophys. Res.*, 96:6435–6457, 1991.

[3] J. C. Clark, E. E. Brabb, H. G. Greene, and D. C. Ross. Geology of Point Reyes penninsula and implications for San Gregorio Fault history. In J. K. Crouch and S. B. Bachman, editors, *Tectonics and Sedimentation Along the California Margin*, pages 67–85. Pacific Section, Soc. Economic Paleontologists and Mineralogists, 1984.

[4] K. J. Coppersmith and G. B. Griggs. Morphology, recent activity, and seismicity of the San Gregorio fault zone. Spec. Rep. 137, Calif. Div. Mines Geol., 1978. in San Gregorio-Hosgri Fault Zone, California, edited by E. A. Silver and W. R. Normark.

[5] J. K. Crouch, S. B. Bachman, and John T. Shay. Post-miocene compressional tectonics along the central California margin. In J. K. Crouch and S. B. Bachman, editors, *Tectonics and Sedimentation Along the California Margin*, pages 37–54. Pacific Section, Soc. Economic Paleontologists and Mineralogists, 1984.

[6] C. DeMets, R. G. Gordon, S. Stein, and D. F. Argus. A revised estimate of Pacific-North America motion and implications for Western North America plate boundary zone tectonics. *Geophys. Res. Lett.*, 14:911–914, 1987.

[7] M. B. Dobrin and C. H. Savit. *Introduction to Geophysical Prospecting.* McGraw-Hill, New York, NY, 4th edition, 1988. 867 pp.

[8] J. P. Eaton. Focal mechanisms of near-shore earthquakes between Santa Barbara and Monterey, California. Open-File Report 84-477, U. S. Dept. Interior Geol. Survey, 1984.

[9] J. Ewing and M. Talwani. Marine deep seismic reflection profiles off central California. *J. Geophys. Res.*, 96:6423–6433, 1991.

[10] Pacific Gas and Electric Company. Diablo Canyon power plant long term seismic program report. Centennial continent/ocean transect #10, Pacific Gas and Electric Company, San Francisco, California, 1988.

[11] W. H. Gawthrop. Seismicity and tectonics of the central California coastal region. Spec. Rep. 137, Calif. Div. Mines Geol., 1978. in San Gregorio-Hosgri Fault Zone, California, edited by E. A. Silver and W. R. Normark.

[12] K. Goodfellow. Special report, geophysical activity in 1989. *The Leading Edge*, 9(11):49–72, 1990.

[13] S. A. Graham and W. R. Dickinson. Apparent offsets of on land geologic features across the San Gregorio-Hosgri fault trend. Spec. Rep. 137, Calif. Div. Mines Geol., 1978. in San Gregorio-Hosgri Fault Zone, California, edited by E. A. Silver and W. R. Normark.

[14] S. A. Graham and W. R. Dickinson. Evidence for 115 kilometers of right slip on the San Gregorio-Hosgri fault trend. *Science*, 199:179–181, 1978.

[15] C. A. Hall. San Simeon-Hosgri fault system, coastal California: Economic and environmental implications. *Science*, 190:1291–1294, 1975.

[16] C. A. Hall, Jr. Origin and development of the Lompoc-Santa Maria pull-apart basin and its relation to the San Simeon-Hosgri strike-slip fault, western California. Spec. Rep. 137, Calif. Div. Mines Geol., 1978. in San Gregorio-Hosgri Fault Zone, California, edited by E. A. Silver and W. R. Normark.

[17] D. H. Hamilton. Characterization of the San Gregorio-Hosgri system, coastal central California. *Geol. Soc. Am. Abstr. Programs*, 19:385–386, 1987. abstract.

[18] D. H. Hamilton and C. R. Willingham. Hosgri fault zone; structure, amount of displacement, and relationship to structures of the western Transverse Ranges. *Geol. Soc. Am. Abstr. Programs*, 19:429, 1977. abstract.

[19] K. L. Hanson, W. R. Lettis, and E. L. Mezger. Late Pleistocene deformation along the San Simeon fault zone near San Simeon, California. *Geol. Soc. Am. Abstr. Programs*, 19:386, 1987. abstract.

[20] J. S. Hornafius. Neogene tectonic rotation of the Santa Ynez Range, western Transverse Ranges, California, suggested by paleomagnetic investigation of the Monterey Formation. *J. Geophys. Res.*, 90:12,503–12,522, 1985.

[21] E. G. Hoskins and J. R. Griffiths. Hydrocarbon potential of northern and central California offshore. *A. A. P. G. Memoir 15*, 1:212–228, 1971.

[22] K. Larner, R. Chambers, M. Yang, W. Lynn, and W. Wai. Coherent noise in marine seismic data. *Geophys.*, 48:854–886, 1983.

[23] J. N. Louie. Imaging of the Hosgri fault, offshore California. *Bull. Seism. Soc. Am.*, 1991. submitted.

[24] B. P. Luyendyk and J. S. Hornafius. Neogene crustal rotations, fault slip, and basin development in southern California. In R. V. Ingersoll and W. G. Ernst, editors, *Cenozoic Basin Development of Coastal California*, pages 259–283. Prentice-Hall, Inc., 1987.

[25] D. S. McCulloch. Regional geology and hydrocarbon potential of offshore central California. In D. W. Scholl, A. Grantz, and J. G Vedder, editors, *Geology and Resource Potential of the Continental Margin of Western North America and Adjacent Ocean Basins—Beaufort Sea to Baja California*, pages 353–401, Houston, Tex., 1987. Circum-Pacific Council for Energy and Mineral Resources.

[26] K. D. McIntosh, D. L. Reed, E. A. Silver, and A. S. Meltzer. Deep structure and structural inversion along the central California continental margin from EDGE seismic profile RU-3. *J. Geophys. Res.*, 96:6459–6473, 1991.

[27] A. S. Meltzer and A. R. Levander. Deep crustal reflection profiling offshore southern central California. *J. Geophys. Res.*, 96:6475–6491, 1991.

[28] J. B. Minster and T. H. Jordan. Vector constraints on Quaternary deformation of the western United States east and west of the San Andreas fault. In J. K. Crouch and S. B. Bachman, editors, *Tectonics and Sedimentation Along the California Margin*, pages 1–16. Pacific Section, Soc. Economic Paleontologists and Mineralogists, 1984.

[29] W. D. Mooney. Introduction to special section on EDGE and related seismic projects, onshore-offshore California. *J. Geophys. Res.*, 96:6421, 1991.

[30] V. S. Mount and J. Suppe. State of stress near the San Andreas fault: Implications for wrench tectonics. *Geology*, 15:1143–1146, 1987.

[31] J. Namson and T. L. Davis. Seismically active fold and thrust belt in the San Joaquin Valley, central California. *Geol. Soc. of Am. Bull*, 100:257–273, 1988.

[32] J. Namson and T. L. Davis. Late cenozoic fold and thrust belt of the southern California Coast Ranges and Santa Maria basin, California. *Am. Assoc. Petroleum Geol. Bull.*, 74:467–492, 1990.

[33] J. B. Saleeby. C-2 Central California Offshore to Colorado Plateau. Centennial continent/ocean transect #10, Geol. Soc. Am., 1986.

[34] W. A. Schneider. Integral formulation for migration in two and three dimensions. *Geophys.*, 43(1):49–76, 1978.

[35] C. L. Seitz. Multicomputers. In C. A. R. Hoare, editor, *Developments in Concurrency and Communication*, pages 131–200. Addison-Wesley, 1990.

[36] E. A. Silver. The San Gregorio-Hosgri fault zone: An overview. Spec. Rep. 137, Calif. Div. Mines Geol., 1978. in San Gregorio-Hosgri Fault Zone, California, edited by E. A. Silver and W. R. Normark.

[37] W. S. Snyder. Structure of the Monterey Formation: Stratigraphic, diagenetic, and tectonic influences on style and timing. In R. V. Ingersoll and W. G. Ernst, editors, *Cenozoic Basin Development of Coastal California*, pages 321–347. Prentice-Hall, Inc., 1987.

[38] J. E. Vidale. Finite-difference calculation of travel times. *Bull. Seism. Soc. Am.*, 78:2062–2076, 1988.

[39] H. C. Wagner. Marine geology between Cape San Martin and Point Sal, south-central California offshore. OFR 74-252, U. S. G. S., 1974.

[40] G. E. Weber and K. R. LaJoie. Late Pleistocene and Holocene tectonics of the San Gregorio fault zone between Moss Beach and Point Ano Nuevo, San Mateo County, California. *Geol. Soc. Am. Abstr. Programs*, 9:524, 1977. abstract.

[41] O. Yilmaz. *Seismic Data Processing*, volume 2 of *Investigations in Geophysics*. Society of Exploration Geophysicists, Tulsa, OK, 1987.

[42] M. D. Zoback, M. L. Zoback, V. S. Mount, J. Suppe, J. P. Eaton, J. H. Healy, D. Oppenheimer, P. Reasenberg, L. Jones, C. B. Raleigh, I. G. Wong, O. Scotti, and C. Wentworth. New evidence on the state of stress of the San Andreas fault system. *Science*, 238:1105–1111, 1987.

Appendix A

Glossary of Terms

automatic gain control (AGC) A means of equalizing amplitudes over the length of a seismic trace. Generally the amplitudes are scaled by the average within a sliding window.

CMP Abbreviation for common midpoint (see **gather**).

deconvolution A process by which the effect of a linear filter is removed from a time series. Spiking deconvolution is designed to remove the source wavelet from the recorded trace, thereby increasing temporal resolution. Predictive deconvolution involves the application of an operator designed to remove the effects of near surface multiples.

F-K filter A filter, generally operating in the frequency-horizontal wavenumber domain, designed to attenuate arrivals with a certain range of apparent velocities. Also called a **dip filter**.

fold The number of traces in a common midpoint gather.

gather A collection of seismic traces with a particular acquisition parameter in common. There are four general types of gathers: **1) shot (or source),** in which all of the traces are recordings of energy from a particular seismic source position; **2) receiver,** in which all of the traces were recorded at a particular survey position; **3) midpoint,** in which all of the traces have in common the surface position half-way between their source and receiver; **4) offset,** in which all of the traces have the same source-receiver distance.

geophone An instrument designed to convert seismic energy into electrical signals.

message-passing A means of communication between nodes on a multicomputer.

migration (or imaging) A means of inverting seismic data recorded as a function of surface position and time, to form an image of reflectivity as a function of surface position and depth. Post-stack migration treats a stacked section as a zero-offset wavefield to perform the inversion: $S(x,t) \Longrightarrow I(x,z)$. Prestack migration inverts the unstacked data: $S(s,g,t) \Longrightarrow I(x,z)$, where s and g are source and geophone positions, respectively.

multicomputer A computational system made up of multiple small computers, called nodes, that work as a group to perform computations, and that communicate by passing messages to one another over a communications network (see Seitz [35]). The grain size of a multicomputer refers to the relative size and power of the nodes–medium-grained nodes fit on a single board, and are approximately the equivalent of a workstation in terms of memory capacity and computational speed. Multicomputers may be homogeneous, in which case all

of the nodes are identical, or heterogeneous, with a variety of specialized nodes.

multiple An arrival of seismic energy that has been reflected more than once, as distinguished from a primary reflection which is the desired product of a reflection survey. A **water-bottom multiple** is energy that reflects from the seafloor to the surface of the ocean, is reflected back to the seafloor, and then reflected back to the receiver.

mute A means of excluding or diminishing seismic energy as a function of offset and time. Often used to remove direct arrivals and surface waves on shot gathers. Also: **stretch mute** by which energy, whose frequency content has been lowered beyond some limit by the normal moveout correction, is removed.

NMO Abbreviation for "normal moveout."

node A computational element in a multicomputer.

normal moveout (NMO) The change in arrival time of energy, returning from a flat-lying reflector, as a function of offset.

offset The distance between the source and receiver in a seismic survey.

roll in, roll out The tapering off of midpoint fold at the beginning and end of a survey.

stack A seismic section intended to simulate a zero-offset section by correcting midpoint gathers for normal moveout and summing the traces together.

statics A constant time shift applied to a seismic trace to correct for elevation and near-surface velocity variations due to weathering.

trace The basic unit of a seismic reflection survey; a time series recorded by a geophone (or hydrophone) after a source of seismic energy has been activated.

Appendix B

Example Program

Below is a fragment of a main program for a computational node, and a stylized function *stack()*. The main program registers *stack()* and turns control over to the notifier. The function *stack()* performs a simplified CMP stack. Since the purpose of this function is to illustrate the use of the trace manager and display manager functions from a computational process, much of the complexity has been relegated to other functions and details have been omitted. The functions themselves may or may not reflect the actual current working versions of the functions, but are illustrative in any case.

```
#include <stdlib.h>
#include <local.h>
#include <sis.h>
        /* etc. */
        .
        .
        .

int stack(int, int *);          /* prototype for stack() */
```

```
        .
        .
        .

int main(int ac, char **av) {

        /* initialization, allocation, etc. */
        .
        .
        .

        /*
         * Register "stack()" with the function id "STACK".
         * Make stack() responsive to the databases with id's
         * "VELOCITY_DB", and "STACK_DB".  The trailing "0"
         * terminates the list of id's
         */

        register_func(stack, STACK, VELOCITY_DB, STACK_DB, 0);

        .
        .
        .

        /* turn control over to the notifier */

        node_main_loop();

        return 0;                   /* exit ISIS */
}

int stack(int n, int *list) {

        int i, ntmax;
        float q;
        float *data, *stack_trace;
        struct usrinfo hdr;
        struct stack_param stk;
        REQUEST w;
```

```
/* first see what's changed by checking the list */

for(i = 0; i < n; i++) {
        switch(list[i]) {
                case VELOCITY_DB:
                        update_velocity_model();
                        break;
                case STACK_DB:
                        get_new_stack_parameters(&stk);
                        break;
                default:
                        err("This shouldn't happen...");
                        return -1;
        }
}

/*
 * get_ntmax() returns the maximum number of time points in
 * any trace in the survey.
 */

ntmax = get_ntmax();

/* allocate a buffer for incoming data */

if((data = (float*)malloc(ntmax * sizeof(float))) == NULL) {
        err("out of memory");
        return -1;
}

/* allocate a buffer for the stack trace */

if((stack_trace = (float*)malloc(ntmax * sizeof(float))) == NULL) {
        err("out of memory");
        return -1;
}

/*
 * getrequest() fills in the default request structure based
 * on the type of request, in this case "MIDLINE" specifies
 * that the request will be for a midpoint gather
```

```
 */

getrequest(MIDLINE, &w);

/*
 * Fill in the request structure.  The "stk" structure
 * contains information from the user interface on the
 * geometry of the stack to be performed.
 */

w.iline = stk.line; /* the line along which the stacking is done */

/*
 * setdataops instructs the trace manager to preform certain
 * operations on the data.  In this case it will perform NMO
 * and mutes based on the current velocity and mute models.
 */

setdataops(DO_NMO | DO_MUTES);

/*
 * Loop over midpoints.
 * q is the distance along the line "stk.line", "stk.q0" is
 * beginning of the section, "stk.q1" is the end of the
 * section, and "stk.dq" is the binning interval.
 */

for(q = stk.q0; q < stk.q1; q += stk.dq) {

        /*
         * Request the midpoint gather "q" units
         * along the line.
         */

        w.MIDPAR.q = q;

        datarequest(&w);

        /* Zero stack_trace for each midpoint. */

        memset((void*)stack_trace, 0, ntmax * sizeof(float));
```

```
        /* Loop over traces in the gather. */

        while(getdata(&hdr, data)) {

                /*
                 * traces come with NMO applied, so we just
                 * have to sum them into the stack!
                 */

                for(i = 0; i < hdr.nt; i++)
                        stack_trace[i] += data[i];
        }

        /*
         * Plot the stack trace.  "PLT_SEC" instructs the
         * display manager that the plot is a section.
         */

        hdr.offset = q;
        draw_trace(stack_trace, &hdr, PLT_SEC);
}

/*
 * Plot the section.  "SPLIT_SCREEN" informs the display
 * manager that this is one segment of a multi-segment image.
 * "REPLACE_IMG" instructs it to completely overwrite the
 * existing image with the new one.
 */

plot_data(SPLIT_SCREEN, REPLACE_IMG);

return 0;        /* return control to the notifier */
```