IMAGE GUIDED BLENDED NEAR NEIGHBOR INTERPOLATION AND ITS APPLICATIONS IN OIL EXPLORATION

By

Hameer Abbasi
Aabgeenay Khan
Rida Maamoor
Imran Azeem

Supervisor:

Brig (R) Khan Minhas

Submitted to the Faculty of Department of Electrical Engineering,

Military College of Signals, National University of Sciences and Technology, Islamabad in partial fulfilment for the requirements of a B.E. Degree in Telecom Engineering

JUNE 2014

ABSTRACT

IMAGE GUIDED BLENDED NEAR NEIGHBOR INTERPOLATION AND ITS APPLICATIONS IN OIL EXPLORATION

A number of interpolation techniques exist that are designed to interpolate data that is sparse in nature over a field to the entire range. These include schemes such as Inverse-Distance Weighting (IDW), Kriging and others. The limitation to these schemes is that they depend only on the location and value of the sparse points.

In some applications, however, the need for a guiding function that is dense in nature is felt. One application, which happens to be the one we are working on, is the guidance of well data (sparse) by seismic date (dense). This is needed because the location of wells alone is insufficient to determine the orientation of layers – the collected seismic data is needed to be able to "guide" the prediction of the layers. As a result, a new technique is needed to be able to meet these requirements.

We propose a technique known as Image Guided Blended Near-Neighbour Interpolation, and aim to define it mathematically, as well as to introduce an algorithm for its fast computation. The properties of well data that can be interpolated include velocity, density, temperature, viscosity and others.

This technique would not only have a computational advantage over existing methods but it would also lead to a reduction in time and cost giving it an edge over all previously known techniques.

CERTIFICATE OF CORRECTNESS AND APPROVAL

It is certified that the work contained in the thesis titled " Image Guided Blended Near Neighbour Interpolation and its Applications in Oil Exploration", carried out by Hameer Abbasi, Aabgeenay Khan, Rida Maamoor and Imran Azeem under the supervision of Brig (R) Muhammad Khan Minhas in partial fulfilment of Degree of Bachelor of Telecommunication Engineering, is correct and approved.

Approved by

_____

Brig (R) Muhammad Khan Minhas

EE Department

Military College of Signals, Rwp

DECLARATION

No portion of the work presented in this dissertation has been submitted in support of another award or qualification either at this institution or elsewhere.

DEDICATION

In the name of Allah, the Most Merciful, the Most Beneficent

To our parents, without whose support and continuous cooperation, a work of this magnitude would not have been possible.

ACKNOWLEDGEMENTS

We are grateful to ALLAH, who has given us guidance, strength and enabled us to accomplish this task. The journey on this road of success had not been possible without His will, and henceforth, all this success and accomplishment, we owe it to Him, in totality.

We are also grateful to our parents and family and well-wishers for their untiring support. We would like to thank our supervisor Brig (R) Muhammad Khan Minhas and co-supervisor Dr. Zaki Murtaza, for their help and motivation throughout the course of our project. Without their help we would have not been able to accomplish anything. Under their guidance, we were able to achieve our goals.

In addition, we'd like to thank our industry sponsors, LMKR, for their guidance and cooperation during the course of this project. We owe the test data and some of the equipment used to complete this work to them.

CONTENTS

TABLE OF FIGURES

Chapter 1 INTRODUCTION

1.1 Introduction

This chapter provides an overview of the project, project motivation, project objectives and organization of the document.

The project entails something like this. The seismic data is present in the form of a volume, which is dense, but not very accurate. Then there is well log data, which is accurate, but sparse. How can the well log data be interpolated that it honours the local structure present in the seismic volume? It turns out that there is a really simple way to do this. Nearest neighbour interpolation; what is in this method is that for every point in the seismic volume, the Euclidean distance in 3-D space to every well log point is calculated. Then simply the nearest neighbour's value at this sample point is written here. Obviously if the implementation isn't efficient, even this naive brute force method will take forever.

Uniformly sampled images are often used to interpolate other data acquired more sparsely with an entirely different mode of measurement. For example, downhole tools enable geophysical properties to be measured with high precision near boreholes that are scattered spatially, and less precise seismic images acquired at the earth's surface are used to interpolate those properties at locations far away from the boreholes.

Image-guided interpolation is designed specifically to enhance this process. Most existing methods for interpolation require distances from points where data will be interpolated to nearby points where data are known.

Image-guided interpolation requires non-Euclidean distances in metric tensor fields

that represent the coherence, orientations and shapes of features in images. This requirement leads to a new method for interpolating scattered data that is called blended neighbour interpolation.

1.2 Motivation

The motivation for implementation of image guided blended near neighbour interpolation is due to increasing need of the existence of an efficient algorithm that could provide correct whereabouts of the location of oil under the seismic layers. During the data-collection phase the well log and the seismic data is collected. Well log data is the one that is sparse in nature and seismic data is one that is dense in nature. The project has been provided by LMKR as a component of one their Geo Graphix department.

The collection and provision of data is basically being done by them. After that by using this data, sparse data is interpolated over dense data. Since the preexisting techniques included near neighbour interpolation and interpolation by PDEs, which were not only computationally inefficient but their complexity used to increase with increasing number of points.

The blended near neighbour interpolation was then thought of as an alternative to these time costing techniques. It not only would lead to reduction in complexity while working with large seismic data but also since the image guided blended near neighbour interpolation is to be used, the resultant seismic images would be easier to interpret than the unguided blended near neighbour interpolation.

1.3 Objectives

This section provides the details of the academic and application objectives of the project. The main goal is the design and implementation on Image Guided Blended Near-Neighbour interpolation in a computationally efficient manner on a variety of platforms including Octave, CPU (C#), GPU (CUDA), as well as FPGAs. It is intended to develop only a library that can be used in an existing software application, not a GUI, as the main purpose was the processing of data.

1.3.1 Academic Objectives

The primary objective was to Implementation of PDEs (Partial Differential Equations) for interpolation. Along with it implementation of numerical methods for solving partial differential equations was also one of the primary objectives. To port the solution of the PDEs and mathematical algorithms into machine implementation were the secondary objectives. .

1.3.2 Application Objectives

It provides more well-organized and proficient algorithm for interpolating sparse well log data over dense seismic data. The main applications include velocity modelling, fault detection and improved reconstruction of seismic images

1.3.2.1 Velocity Modelling

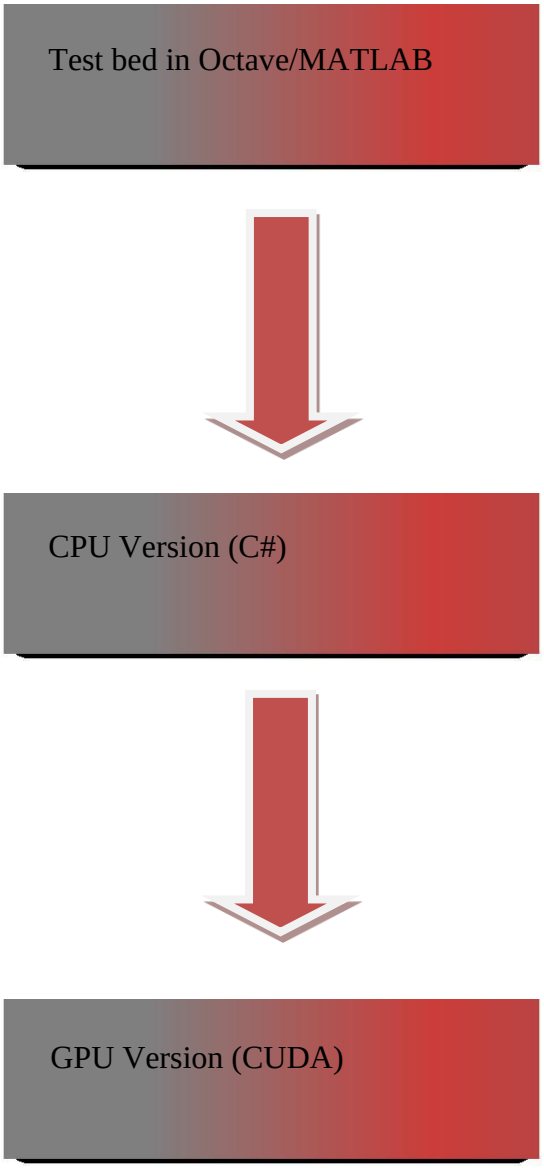
Velocity is a fundamental property of rocks that depends on density and elastic moduli.

It varies laterally as well as vertically due to the physical-geologic variations of rocks. Velocities are determined less precisely from closely spaced seismic reflection data, while precise measurements are obtained from widely spaced boreholes. Velocities appear to be complex due to their various types and usage in processing and interpretation applications. Velocities are measured directly and more precisely by using borehole techniques such as sonic logs, check shots, and vertical seismic profiling.

Although seismic is the main source of velocity distribution information for the whole survey area, borehole velocities at widely dispersed well points provide reliable correlation and control for calibrating the seismic velocities. The velocity model is then developed to understand the true nature of the seismic data.

1.4 Block Diagram

The basic block diagram of the project can be represented below. Here each block is explained and correlated to the corresponding block till reaches the final output.


Test bed in Octave/MATLAB
CPU Version (C#)
GPU Version (CUDA)

First of all the test development had been done in Octave/ MATLAB which is an open ware. After successful implementation in it, the single thread version is first developed in C# to check timing optimization, since a 3d version could not be implemented in MATLAB due to exorbitant amount of time it took (several hours), a more efficient parallel thread version is implemented in C# to speed up processing time.

The velocity model as required by LMKR had to be a final algorithm developed at GPU having much more processing speed than the C# or other versions.

1.5 Organization of Document

Chapter 2 is about the literature review. Chapter 3 explains system modules particularly the hardware used in the project. Chapter 4 consists of detail discussion of the project design and the development stages. In Chapter 5 details of project testing and evaluation are given, also the project limitations are defined. Chapter 6 deals with future works and applications related to Implementation of image guided blended near neighbour interpolation.

The codes cannot be disclosed on account of the non-disclosure agreement that has been signed with LMKR. In the end is bibliography.

Chapter 2 LITERATURE REVIEW

2.1 Spatial Interpolation

Interpolation is based on the assumption that spatially distributed objects are spatially correlated; in other words, things that are close together tend to have similar characteristics.

There are two main groupings of interpolation techniques: deterministic and geostatistical. Deterministic interpolation techniques create surfaces from measured points, based on either the extent of similarity (inverse distance weighted) or the degree of smoothing (radial basis functions). Geostatistical interpolation techniques (kriging) utilize the statistical properties of the measured points. Geostatistical techniques quantify the spatial autocorrelation among measured points and account for the spatial configuration of the sample points around the prediction location.

Deterministic interpolation techniques can be divided into two groups, global and local. Global techniques calculate predictions using the entire dataset. Local techniques calculate predictions from the measured points within neighbourhoods, which are smaller spatial areas within the larger study area. Geostatistical Analyst provides global polynomial as a global interpolator and inverse distance weighted, local polynomial, radial basis functions, kernel smoothing, and diffusion kernel as local interpolators.

A deterministic interpolation can either force the resulting surface to pass through the data values or not. An interpolation technique that predicts a value that is identical to the measured value at a sampled location is known as an exact interpolator. An inexact interpolator predicts a value that is different from the measured value.

Spatial and spatial-temporal distributions of both physical and socioeconomic phenomena can be approximated by functions depending on location in a multi-dimensional space, as multivariate scalar, vector, or tensor fields. Typical examples are elevations, climatic phenomena, soil properties, population densities, fluxes of matter, etc. Visualization, analysis, and modelling are usually based on a raster representation.

Moreover, the phenomena can be measured using various methods (remote sensing, site sampling, etc.) leading to heterogeneous data sets with different digital representations and resolutions which need to be combined to create a single spatial model of the phenomenon under study.

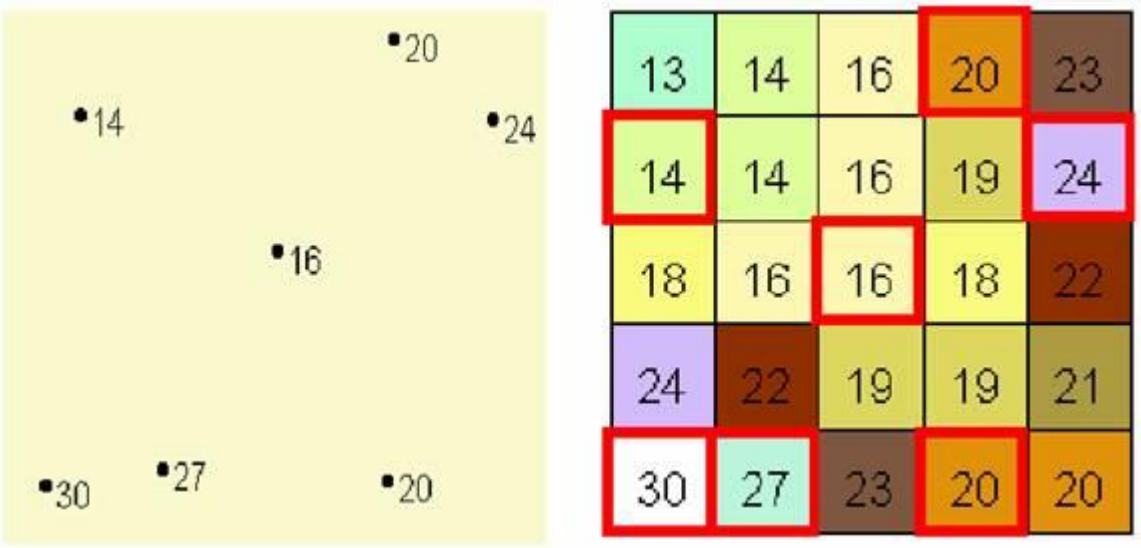


Figure 2-1: Simple discrete 2D interpolation.

On left is point set data of known values. On the right is raster interpolated from these points.

The following methods have been employed to have interpolation of 2D scattered data.

2.2 PDE Method for Interpolation

Partial differential equations (PDEs) have recently shown to be very promising for image interpolation and compression.

The Delaunay triangulation of a set of points P in a plane is a non-overlapping triangulation such that no point in P lies in the circumcircle of any triangle. Delaunay triangulations are often used to construct meshes used in finite-element or finite-volume numerical simulations of partial differential equations.

For a set of points on the same line there is no Delaunay triangulation (the notion of triangulation is degenerate for this case). For four or more points on the same circle (e.g., the vertices of a rectangle) the Delaunay triangulation is not unique: each of the two possible triangulations that split the quadrangle into two triangles satisfies the "Delaunay condition", i.e., the requirement that the circumcircles of all triangles have empty interiors.

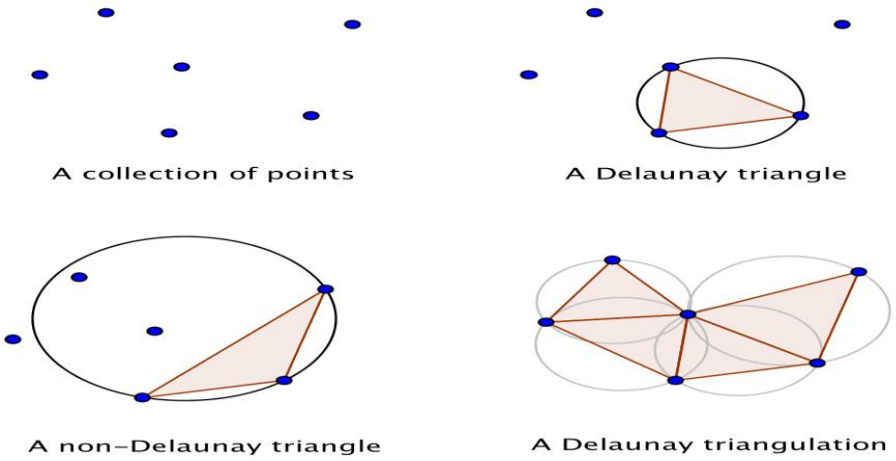


Figure 2-2: Delaunay triangles and Delaunay triangulation.

2.2.1 **Radial basis functions**

RBFs are means to approximate multivariable (also called *multivariate*) functions by linear combinations of terms based on a single univariate function.

The structure of an RBF networks in its most basic form involves three entirely different layers

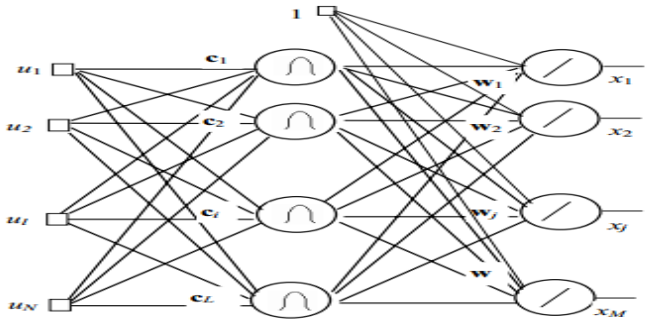


Figure 2-3: Flow graph for radial basis functions.

Their main features are:

1. They are two-layer feed-forward networks.

2. The training/learning is very fast.

3. The networks are very good at interpolation

But there are two serious problems with these exact interpolation networks:

1. They perform poorly with noisy data

2. They are not computationally efficient

Sums of radial basis functions are typically used to approximate given functions. This approximation process can also be interpreted as a simple kind of neural network. RBFs are also used as a kernel in support vector classification. Radial Basis functions are used to interpolate scattered data in two or more dimensions. Several

applications in graphics, geophysics, and learning use interpolation methods based on RBFs.

One issue with using the technique for large data sets is that direct solution of the interpolation problem for N points scales as $O(N^3)$ while a single evaluation of the radial basis function fit at a point itself requires $O(N)$ operations.

2.3 Natural Neighbour Interpolation

Natural neighbour interpolation finds the closest subset of input samples to a query point and applies weights to them based on proportionate areas to interpolate a value.

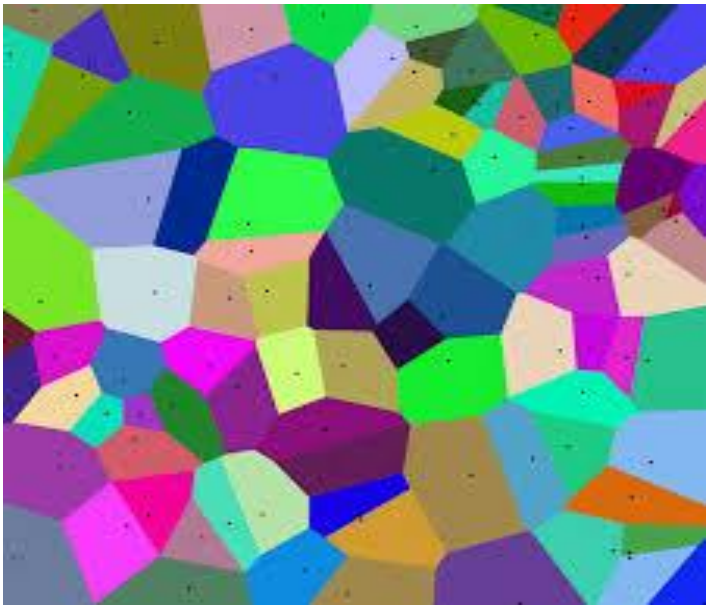


Figure 2-4: Nearest neighbour interpolation.

It is also known as Sibson or "area-stealing" interpolation. Its basic properties are that it's local, using only a subset of samples that surround a query point, and interpolated heights are guaranteed to be within the range of the samples used. It does not infer

trends and will not produce peaks, pits, ridges, or valleys that are not already represented by the input samples. The surface passes through the input samples and is smooth everywhere except at locations of the input samples. Break lines may be used, in the case of TIN to raster interpolation, to augment the surface, creating linear discontinuities where appropriate such as along roadsides and water bodies. It adapts locally to the structure of the input data, requiring no input from the user pertaining to search radius, sample count, or shape. It works equally well with regularly and irregularly distributed data.

The natural neighbours of any point are those associated with neighbouring Voronoi (Thiessen) polygons. Initially, a Voronoi diagram is constructed of all the given points, represented by the olive-coloured polygons. A new Voronoi polygon, beige colour, is then created around the interpolation point (red star). The proportion of overlap between this new polygon and the initial polygons are then used as the weights.

By comparison, a distance-based interpolator, such as inverse distance weighted (IDW), would assign similar weights to the northernmost point and to the north-eastern point based on their similar distance from the interpolation point. Natural neighbour interpolation, however, assigns weights of 19.12% and 0.38%, respectively, which are based on the percentage of overlap.

Nearest neighbour interpolation is well defined for only those points $\vec{x}$ with a single nearest sample point $\vec{x}_k$. At points $\vec{x}$ that are equidistant from two or more sample points, the nearest neighbor interpolant $p(\vec{x})$ is discontinuous and undefined.

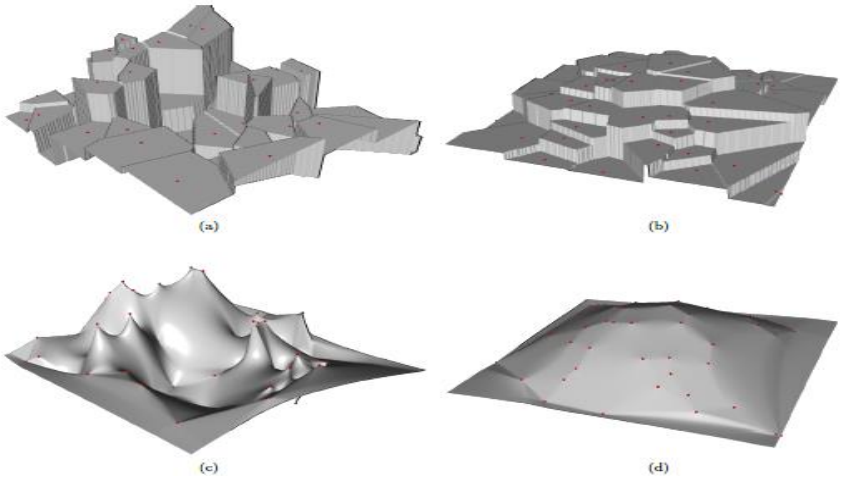


Figure 2-5: Nearest neighbour and natural neighbour.

2.4 Blending Nearest Neighbour

Discrete approximation of the blending equation yields a large sparse system of equations that are best solved by an iterative method. The number of simple conjugate-gradient iterations required to converge to a solution is roughly proportional to the maximum distance $\mathrm{d}(\vec{x})$. For any dimension n of points $\vec{x} \in \mathbb{R}^n$, the computational complexity of discrete blended neighbour interpolation grows only linearly with distance. For sample points scattered at large distances, this linear dependence contrasts favourably with the quadratic dependence in 2D (or cubic dependence in 3D) for discrete natural neighbour interpolation

The first step is to compute for all samples the distance to the nearest known sample and the value of that known sample. This first step produces a distance map and a nearest-neighbour interpolant.

The second step is to blend (smooth) the nearest-neighbour interpolant, where the extent of smoothing varies spatially and is proportional to distances in the distance map.

In tensor-guided gridding, we replace distance with time. Time is a simple term for non-Euclidean distance measured in a metric-tensor field. So "nearest" now means nearest in time. In the first step we compute a time map by solving an eikonal equation with coefficients that may be both anisotropic and spatially varying. In the second step, we blend the nearest-neighbour interpolant with an anisotropic and spatially varying smoothing filter.

The default tensor field is homogeneous and isotropic. In this special case, time is equivalent to distance, and tensor-guided gridding is similar to gridding with Sibson's natural neighbour interpolant.

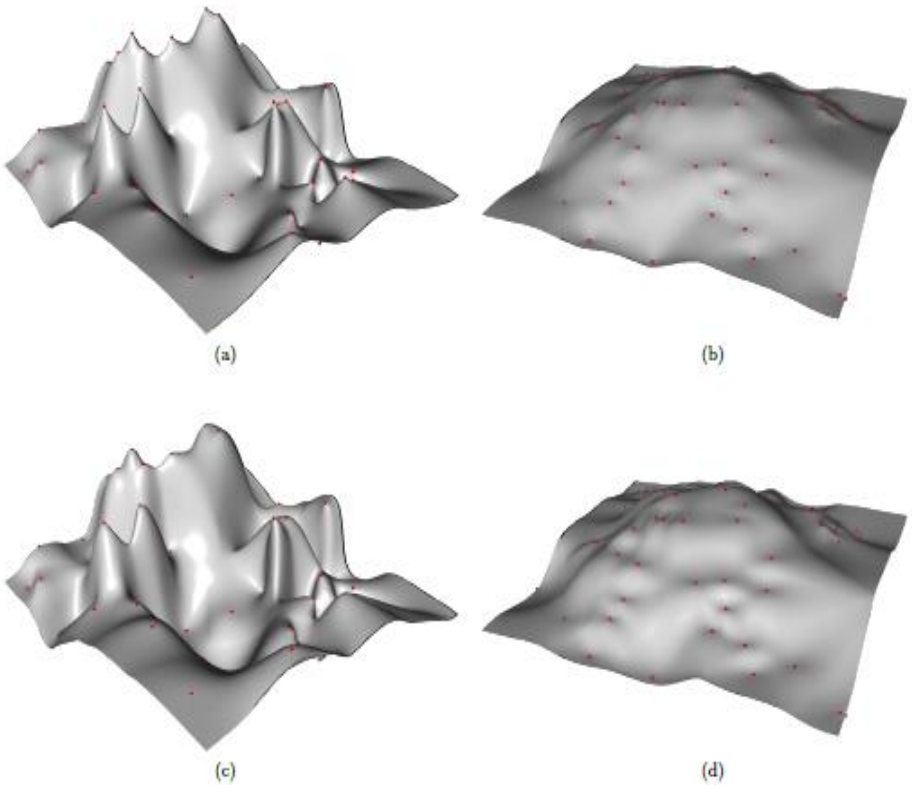


Figure 2-6: Blending nearest neighbour

2.5 Blending Nearest Neighbour using Tensor Fields:

The tensor represents the coherence, orientation, and dimensionality of features in the image that will guide interpolation. Intuitively, this tensor field alters interpolation so that known sample values within spatially coherent image features are given more weight than values on opposite sides of such features or where the image is less coherent. For both images, we will compute the displayed tensor fields $\mathbf{D}(\vec{x})$ from a structure tensor field $\mathbf{S}(\vec{x})$.

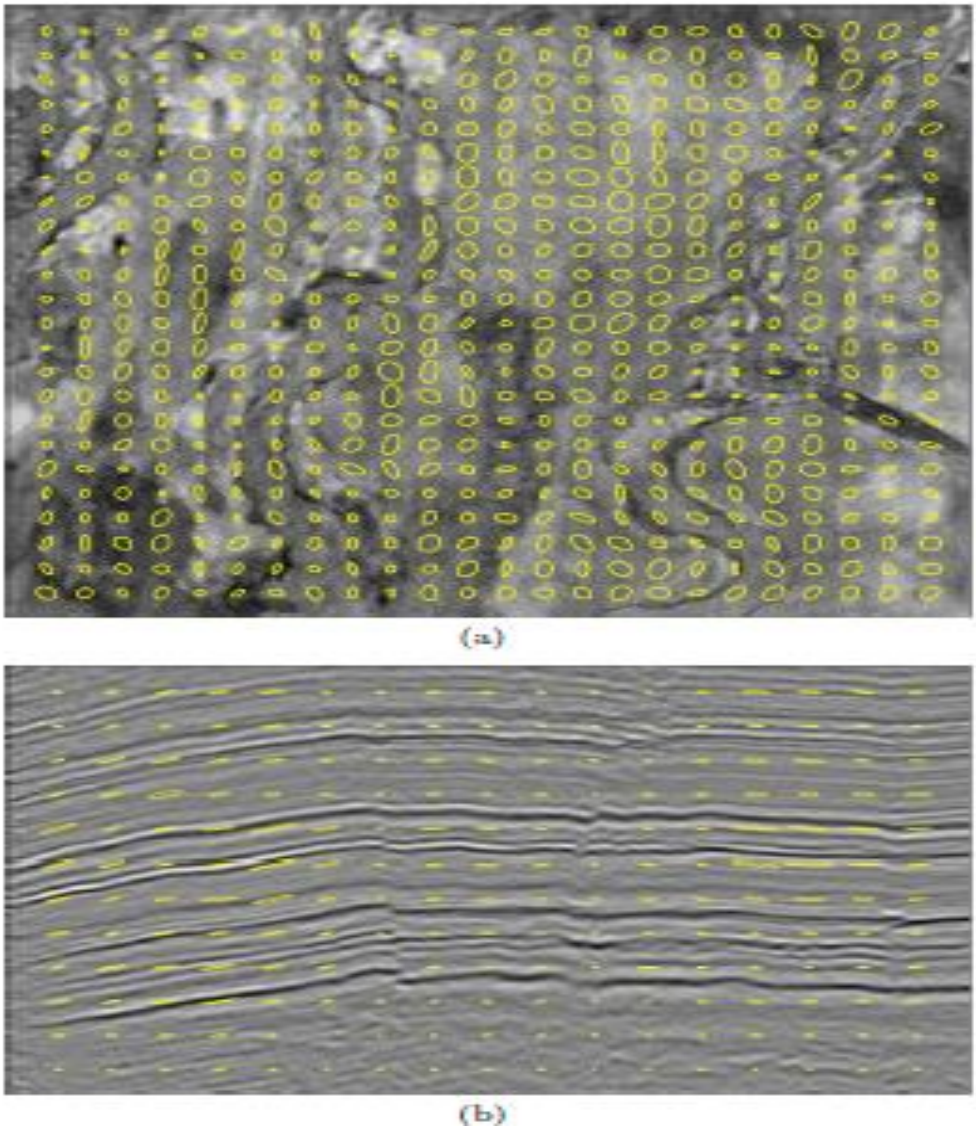


Figure 2-7: Visual representation of structure tensors.

Assuming that an image is available, the accuracy of image-guided interpolation depends on the extent to which the property being interpolated is correlated with image features. If no such correlation exists, then a simpler and faster image-ignorant interpolation may be more accurate than image-guided interpolation with an irrelevant image. In this case, blended neighbour interpolation is an efficient alternative to discrete natural neighbour interpolation. In many contexts, however, an isotropic and constant tensor field is inappropriate.

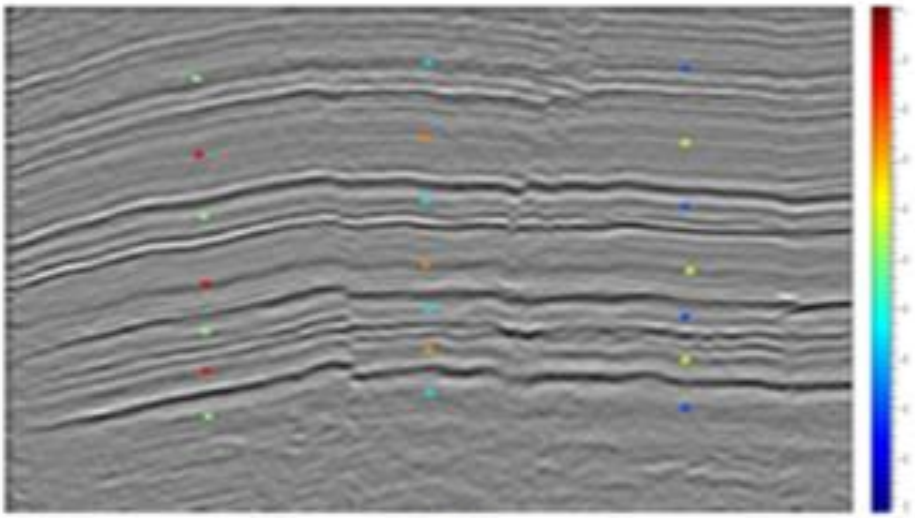
Even when a useful image is unavailable, it may still be possible to construct an anisotropic and inhomogeneous tensor field $\mathbf{D}(\vec{x})$ to guide interpolation. The proposed two-step process might more accurately be called tensor-guided blended neighbour interpolation, because it requires only the tensor field $\mathbf{D}(\vec{x})$, not the image. The tensor fields are derived from uniformly-sampled seismic images, and then interpolated scattered data on the same uniform sampling grid. In some applications, it may be desirable to interpolate scattered data with higher resolution, and nothing in the method prevents this. Image-guided blended neighbour interpolation requires only that we provide a tensor $\mathbf{D}$ for all uniformly sampled locations $\vec{x}$ where we interpolate.

The tensor fields used in image-guided blended neighbour interpolation are analogous to spatial correlation functions (variograms) used in kriging, a geostatistical interpolation method in which subsurface properties are modelled as random variables. But the interpolation methods are otherwise rather different. Where images have less resolution than desired for geo statistical modelling, blended neighbour interpolation might be used to provide image-guided trends for Kriging and other geo

statistical methods. In developing blended neighbour interpolation we retained most of the desirable features of natural neighbour interpolation, but we gave up locality. That is, the blended neighbour interpolant depends on all scattered known samples, even those that are very far way, as finite-difference approximations yield a sparse system of linear equations that we must solve simultaneously.

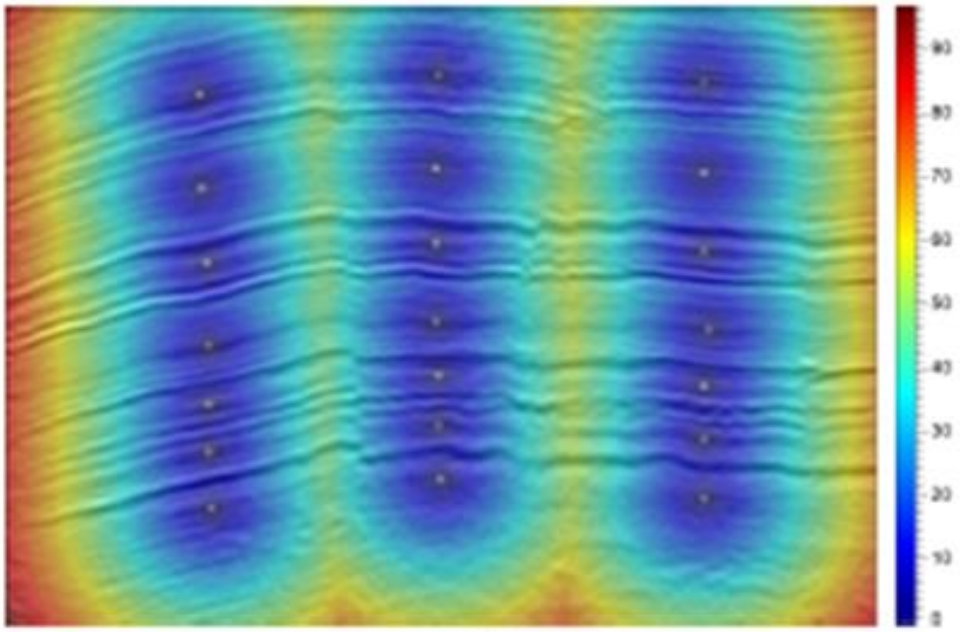
In practice, the neighbourhood in blended neighbour interpolation is quite limited and errors in assuming a local region of influence for each known sample may be less than those due to the use of an iterative solver for the system of blending equations.

In contrast, the cost of solving the blending equations with the simplest conjugate-gradient method grows only linearly with distances or times to nearest known samples. The number of iterations required for such an iterative solver to converge depends in part on the accuracy required in the blended neighbour interpolant. The number of iterations might be reduced by the use of pre conditioners, including multi grid methods.



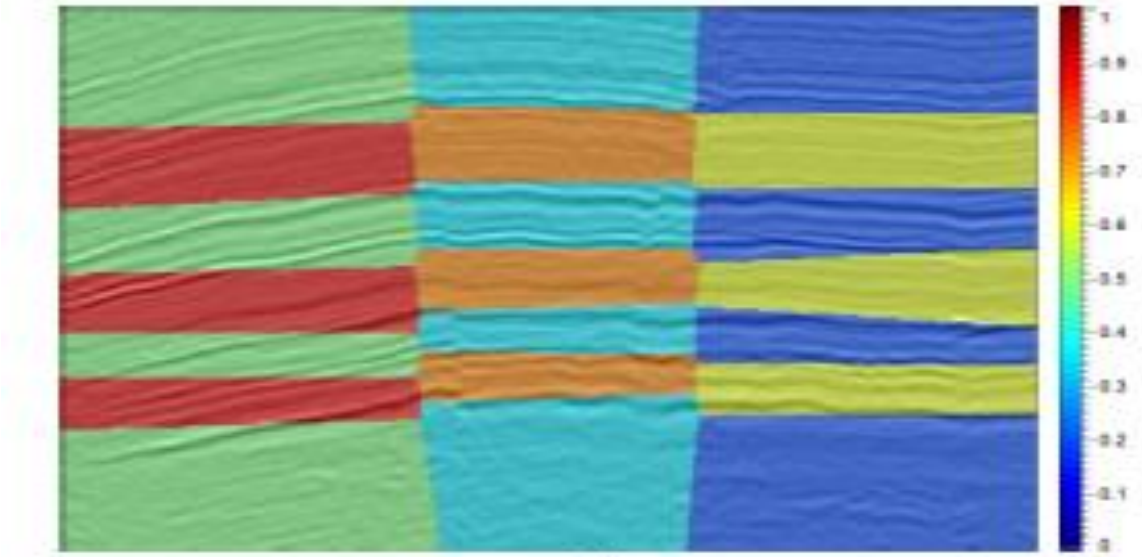
(a)

Figure 2-8: Input image with marked known points.



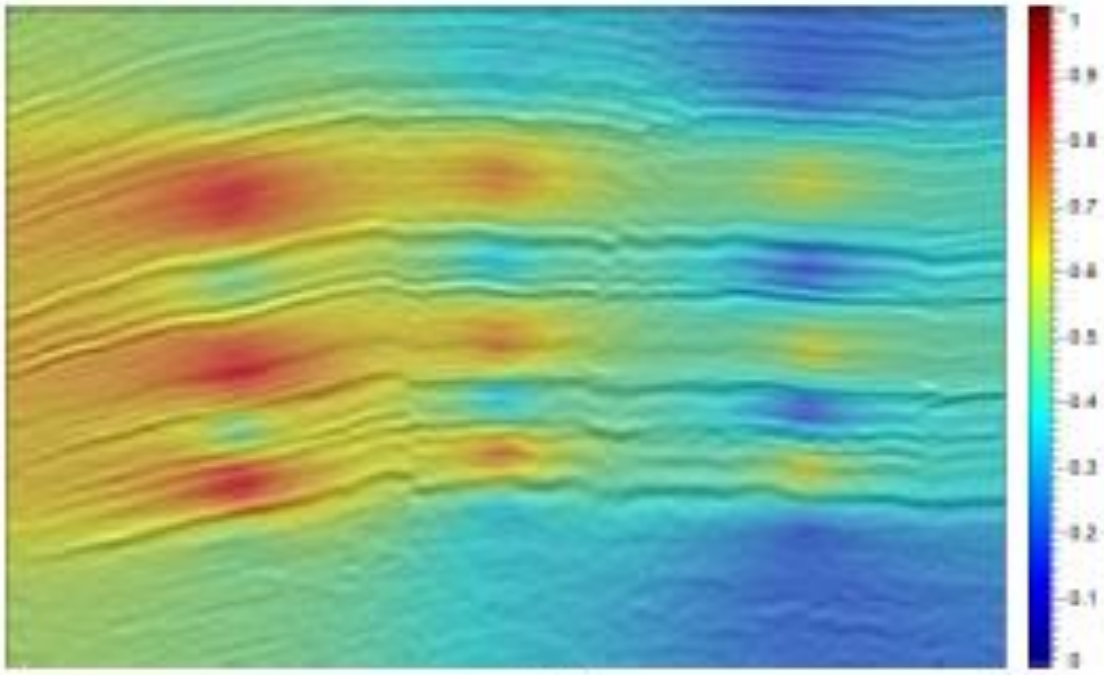
(b)

Figure 2-9: Distance map.



(c)

Figure 2-10: Nearest neighbour interpolant.



(d)

Figure 2-11: Blended nearest neighbour interpolant.

Blended neighbour interpolation of scattered data is similar to the classic method of natural neighbour interpolation, in that both methods smooth a nearest neighbour interpolant, and the extent of smoothing grows with distance to the nearest known sample point. The interpolants are similar but not identical, and the difference between the two methods lies in their smoothing filters. In blended neighbour interpolation a smoothing filter is implied by the solution of a partial differential equation. In natural neighbour interpolation the smoothing filter explicitly computes weighted sums of nearest neighbour sample values. When Euclidean distances are used, the weights in natural neighbour interpolation are simply the areas of polygons, and can be computed efficiently with suitable data structures. However, in non-Euclidean metric tensor fields, these areas must be computed numerically, and for this case blended neighbour interpolation is an efficient alternative to natural neighbour interpolation. In image-guided interpolation we derive metric tensor fields from images, so that the blended neighbour interpolant conforms to image features, while retaining many of the attractive features of the natural neighbour interpolant.

2.6 Chapter Summary

In this chapter it is described in detail how interpolation is being done, its various forms and limitations to various forms. A step by step foundation is the made to form the final outcome why an image guided blended near neighbour interpolation is implemented. Along with it, it is established why a tensor guided blended near neighbour interpolation would be more reliable and easier to understand and follow.

Chapter 3 PROJECT MODULES

3.1 Octave Implementation

GNU Octave is a high-level interpreted language, primarily intended for numerical computations. It provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. It also provides extensive graphics capabilities for data visualization and manipulation. Octave is normally used through its interactive command line interface, but it can also be used to write non-interactive programs, primarily intended for numerical computations. GNU Octave is normally used through its interactive interface (CLI and GUI), but it can also be used to write non-interactive programs.

GNU Octave can do arithmetic for real, complex or integer-valued scalars and matrices, solve sets of nonlinear algebraic equations, integrate functions over finite and infinite intervals, and integrate systems of ordinary differential and differential-algebraic equations.

GNU Octave uses the GNU readline library to handle reading and editing input. By default, the line editing commands are similar to the cursor movement commands used by GNU Emacs, and a vi-style line editing interface is also available. At the end of each session, the command history is saved, so that commands entered during previous sessions are not lost.

The GNU Octave distribution includes a 650+ page Texinfo manual. Access to the complete text of the manual is available via the doc command at the GNU Octave prompt.

Octave runs on various Unices—at least Linux and Solaris, Mac OS X, Windows and anything you can compile it on. Binary distributions exist at least for Debian, SUSE, Fedora and RedHat Linuxes (Intel and AMD CPUs, at least), for Mac OS X and Windows' 98, 2000, XP, Vista, and 7.

Two and three dimensional plotting is fully supported using gnuplot and an experimental OpenGL backend.

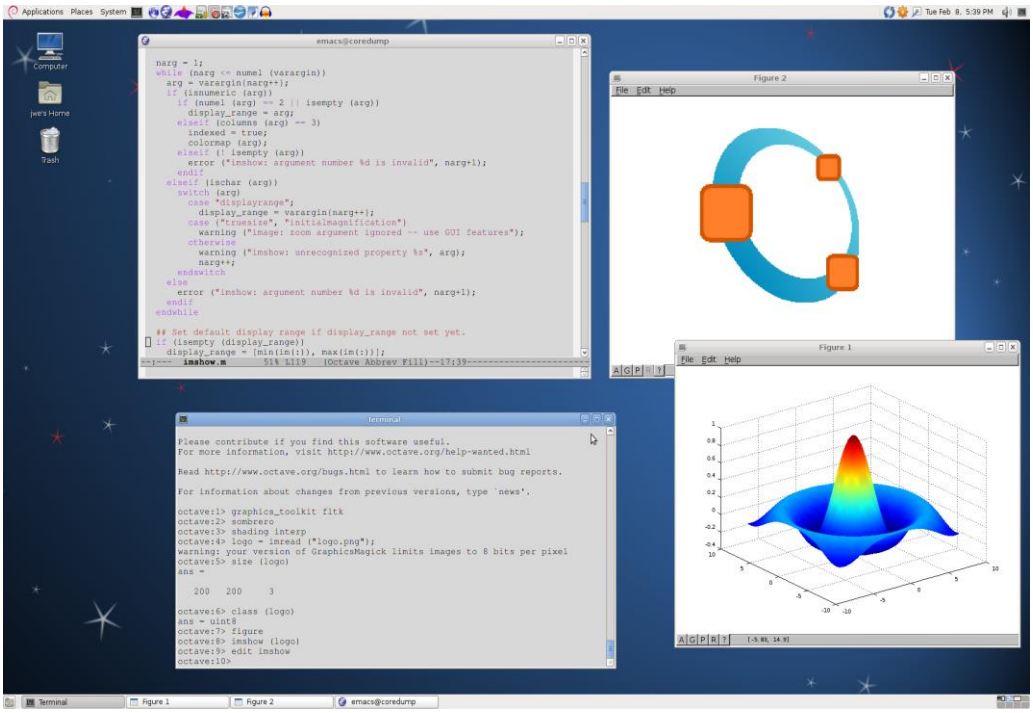


Figure 3-1: Interface for GNU Octave

The underlying numerical solvers are currently standard Fortran ones like LAPACK, LINPACK, ODEPACK, the BLAS, etc., packaged in a library of C++ classes. If possible, the Fortran subroutines are compiled with the system's Fortran compiler, and

called directly from the C++ functions. If that's not possible, you can still compile Octave if you have the free Fortran to C translator f2c.

Octave is also free software. The Octave language is quite similar to MATLAB so that most programs are easily portable. Octave was chosen as a test bed because of its wide range of available libraries and its inter-operability with MATLAB. It was only used for the 2-D model, but as the runtime was approximately 45 minutes, we didn't implement the 3-D model.

The version developed in Octave was very slow. It had a 45-minute runtime for a realistic seismic image with a resolution of 200×251 and 15 known sample files.

This is slow.

3.2 C# Version

Visual C# is modern, high-level, multi-paradigm, general-purpose programming language for building apps using Visual Studio and the .NET Framework. C# is designed to be simple, powerful, type-safe, and object-oriented. The many innovations in C# enable rapid application development while retaining the expressiveness and elegance of C-style languages.

The time-restrictive nature of the Octave implementation led us to develop an optimized version in C# (both 2-D and 3-D). A CPU version was created in C#, for both serial (single-threaded) and parallel (multi-threaded) CPUs. Optimizations such as garbage collection reduction, algorithm simplifications and reductions in the time required to traverse the data structures were applied to the initial implementation. For the multi-threaded version, spin-locks and wait-locks were applied to reduce waiting times. As a result, the time required to process the same image as was processed in

Octave was 2.3 seconds for the single-threaded and 781 milliseconds for the multi-threaded version. Further optimisations are planned for reduction of the memory footprint, but the CPU version can be considered complete and fulfils the requirements set out by LMKR.

For the 3-D version, all the optimisations outlined above were applied in parallel. We only tested the runtime for the multi-threaded version, which turned out to be 1 hour and 45 minutes for realistic sized data with known samples in excess of one thousand.

3.3 GPU Version

A graphics processing unit (GPU), also occasionally called visual processing unit (VPU), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics, and their highly parallel structure makes them more effective than general-purpose CPUs for algorithms where processing of large blocks of data is done in parallel. In a personal computer, a GPU can be present on a video card, or it can be on the motherboard

Figure 3-2: Microchip with nVidia GPU for CUDA.

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model created by NVIDIA and implemented by the graphics processing units (GPUs) that they produce. CUDA gives program developers direct access to the virtual instruction set and memory of the parallel computational elements in CUDA GPUs.

Using CUDA, the GPUs can be used for general purpose processing (i.e., not exclusively graphics); this approach is known as GPGPU

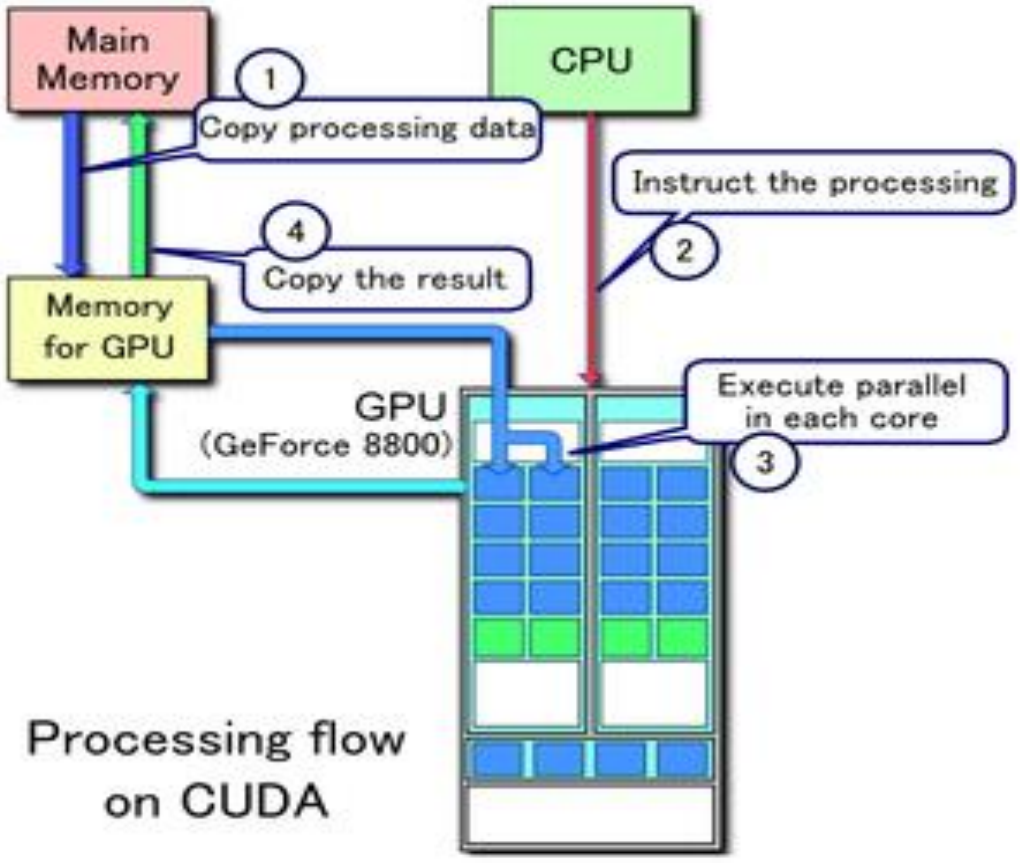


Figure 3-3: CUDA Workflow

Chapter 4 DESIGN AND DEVELOPMENT

This chapter entails the development of an interpolation algorithm that takes into account the structure of the seismic data.

4.1 Input and Output

4.1.1 Input

- A number of of sample points are known. Value of the scalar field at those points is known.
    - A seismic image is provided with values known over the entire field over which to interpolate.

4.1.2 Output

- A scalar field with values defined by the sample points and structure defined by the structure tensor.
- Tensor field calculation.

The equations that are programmed are given as:

4.1.2.1 Step 1

Solve

$$\nabla t(\mathbf{x}) \cdot \mathbf{D}(\mathbf{x}) \nabla t(\mathbf{x}) = 1, \qquad \mathbf{x} \notin \chi$$

$$t(\mathbf{x}) = 0 \qquad \mathbf{x} \in \chi$$

for

$t(\mathbf{x})$: The minimum traveltime from $\mathbf{x}$ to the nearest known sample point $\mathbf{x}_k$, and

$p(\mathbf{x})$: The value f_k corresponding to the sample point $\mathbf{x}_k$ nearest (in traveltime) to the point $\mathbf{x}$. [1] [2]

4.1.2.2 Step 2

$$q(\mathbf{x}) - \frac{1}{e}\boldsymbol{\nabla} \cdot t^2(\mathbf{x})\mathbf{D}(\mathbf{x})\boldsymbol{\nabla} q(\mathbf{x}) = p(\mathbf{x})$$

for the blended neighbour interpolant $q(\mathbf{x})$. [3] [4]

4.2 Tensor Field

Tensors are geometric objects that describe linearity between vectors, scalars, and other tensors. Elementary examples of such relations include the dot product, the cross product, and linear maps. Vectors and scalars themselves are also tensors. A tensor can be represented as a multi-dimensional array of numerical values.
The order (also *degree*) of a tensor is the dimensionality of the array needed to represent it, or equivalently, the number of indices needed to label a component of that array. For example, a linear map can be represented by a matrix, a 2-dimensional array, and therefore is a 2nd-order tensor. A vector can be represented as a 1-dimensional array and is a 1st-order tensor. Scalars are single numbers and are thus 0th-order tensors.

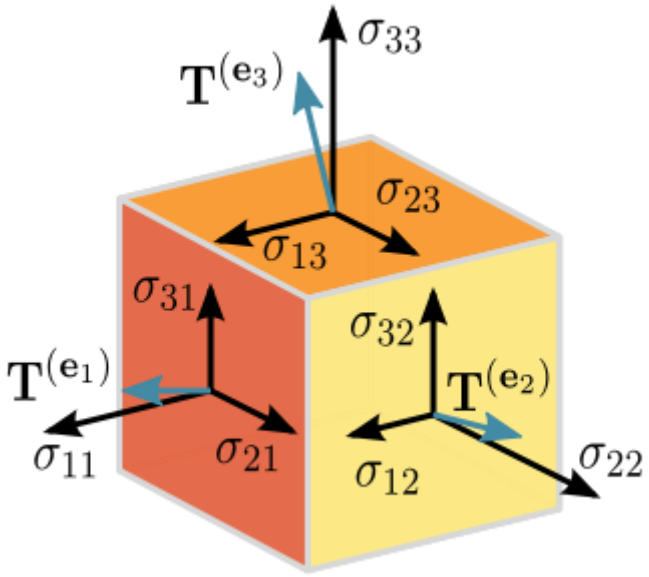


Figure 4-1: Visual representation of a tensor field.

Cauchy stress tensor, a second-order tensor. The tensor's components, in a three-dimensional Cartesian coordinate system, form the matrix

$$\begin{aligned}\sigma &= \left[\mathbf{T}^{(\mathbf{e}_1)}\mathbf{T}^{(\mathbf{e}_2)}\mathbf{T}^{(\mathbf{e}_3)}\right] \\ &= \begin{bmatrix}\sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33}\end{bmatrix}\end{aligned}$$

whose columns are the stresses (forces per unit area) acting on the e_1, e_2, and e_3 faces of the cube.

Tensors are used to represent correspondences between sets of geometric vectors. For example, the Cauchy stress tensor T takes a direction v as input and produces the stress $T^{(v)}$ on the surface normal to this vector for output thus expressing a relationship between these two vectors, shown in the figure (right).

Because they express a relationship between vectors, tensors themselves must be independent of a particular choice of coordinate system. Taking a coordinate basis or frame of reference and applying the tensor to it results in an organized multidimensional array representing the tensor in that basis, or frame of reference.

In mathematics, physics, and engineering, a tensor field assigns a tensor to each point of a mathematical space (typically a Euclidean space or manifold). Tensor fields are used in differential geometry, algebraic geometry, general relativity, in the analysis of stress and strain in materials, and in numerous applications in the physical sciences and engineering. As a tensor is a generalization of a scalar (a pure number representing a value, like length) and a vector (a geometrical arrow in space), a tensor field is a generalization of a scalar field or vector field that assigns, respectively, a scalar or vector to each point of space.

4.3 Fast Marching Algorithm

Fast marching algorithm is employed in solution of the first equation.

The Fast Marching Method solves the general static Hamilton-Jacobi equation, which applies in the case of a convex, non-negative speed function. Starting with an initial position for the front, the method systematically marches the front outwards one grid point at a time, relying on entropy-satisfying schemes to produce the correct viscosity solution. The main idea is exploit a fast heap-sort technique to systematically locate the proper grid point to update, so that one need never backtrack over previously evaluated grid points. The resulting technique sweeps through a grid of N total points

in N log N steps to obtain the evolving time position of the front as it propagates through the grid

The Fast Marching Method makes use of stationary approach to the problem. At first glance, this sounds counter-intuitive; we are going to trade a moving boundary problem for one in which nothing moves at all! To see how this is done, imagine a grid laid down on top of the problem:

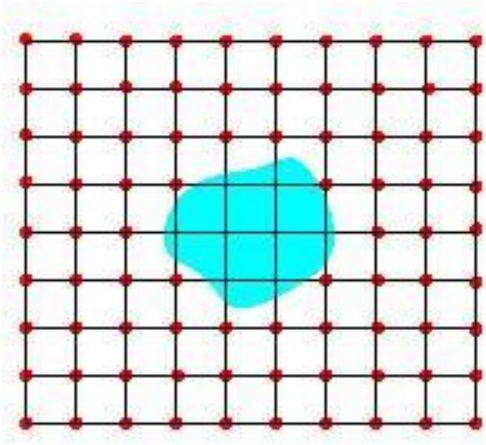


Figure 4-2: Fast marching algorithm.

Suppose that somebody is standing at each red grid point with a watch. When the front crosses each grid point, the person standing there writes down this crossing time T. This grid of crossing time values T(x,y) determines a function; at each grid point T, T(x,y) gives the time at which the front crosses the point (x,y).

As an example, suppose the initial disturbance is a circle propagating outwards. The original region (the blue one on the left below) propagates outwards, crossing over each of the timing spots. The function T(x,y) gives a cone-shaped surface, which is

shown on the right. This surface has a great property; it intersects the xy plane *exactly* where the curve is initially. Better yet, at any height T the surface gives the set of points reached at time T. The surface on the right below is called the arrival time surface, because it gives the arrival time.

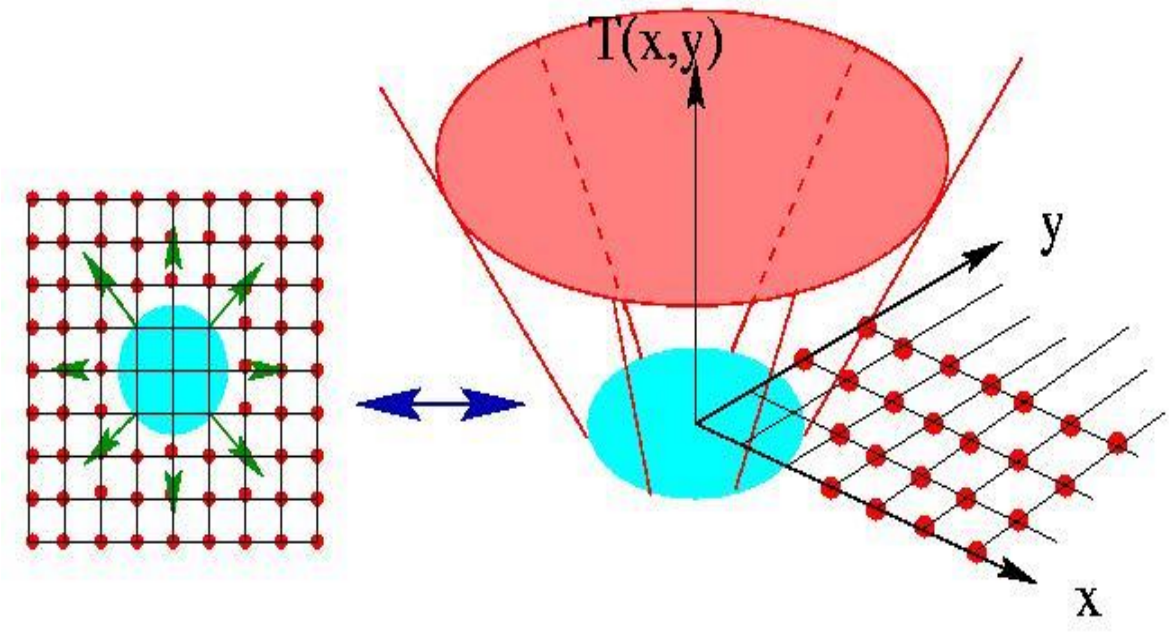


Figure 4-3: Calculated time map and remaining points.

How can this stationary surface be constructed? As a motivation, imagine scaffolding being erected around a house! One stands on one of the boards, puts a board above the head, and then moves to another board at the same level and put a board one level up. Once all the boards are placed at a given level, one then climbs up to the next level set repeat the process. The thing to remember is that the scaffolding is built from the ground up; each level must be completed before the next is begun.

The *Fast Marching Method* imitates this process. Given the initial curve (shown in

red), stand on the lowest spot (which would be any point on the curve), and build a little bit of the surface that corresponds to the front moving with the speed F. Repeat the process over and over, always standing on the lowest spot of the scaffold, and building that little bit of the surface. When this process ends, the entire surface has been built.

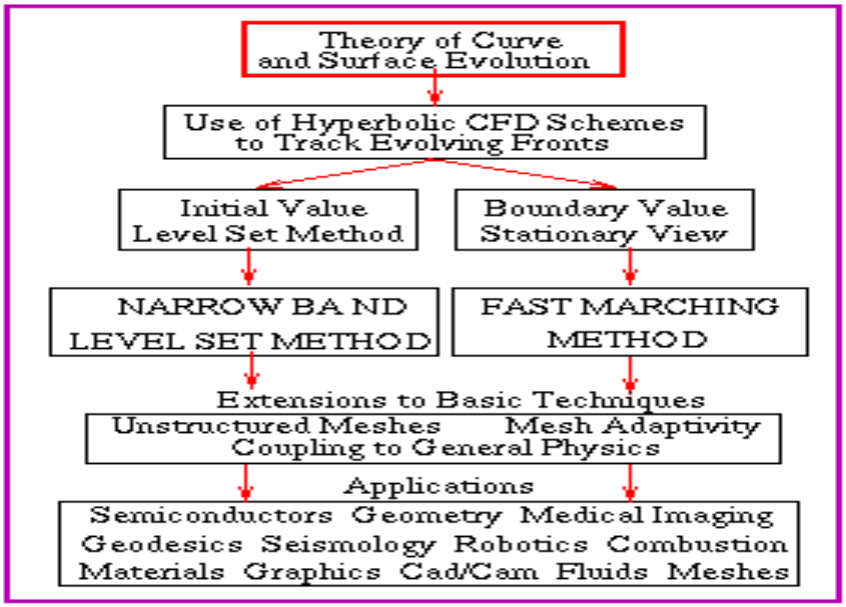


Figure 4-4: Categorization of Fast Marching Algorithms.

4.4 Block wise Conjugate Gradient Method

In mathematics, the conjugate gradient method is an algorithm for the numerical solution of particular systems of linear equations, namely those whose matrix is symmetric and positive-definite. The conjugate gradient method is often implemented as an iterative algorithm, applicable to sparse systems that are too large to be handled by a direct implementation or other direct methods such as the Cholesky decomposition. Large sparse systems often arise when numerically solving partial differential equations or optimization problems.

The conjugate gradient method can also be used to solve unconstrained optimization problems such as energy minimization

Suppose we want to solve the following system of linear equations

$$Ax = b$$

for the vector x where the known *n*-by-*n* matrix A is symmetric (i.e. $A^T = A$), positive definite (i.e. $x^TAx > 0$ for all non-zero vectors x in R^n), and real, and b is known as well.

We denote the unique solution of this system by $\mathbf{x}_*$

The conjugate gradient method can theoretically be viewed as a direct method, as it produces the exact solution after a finite number of iterations, which is not larger than the size of the matrix, in the absence of round-off error. However, the conjugate gradient method is unstable with respect to even small perturbations, e.g., most directions are not in practice conjugate, and the exact solution is never obtained. Fortunately, the conjugate gradient method can be used as an iterative method as it provides monotonically improving approximations $\mathbf{x}_k$ to the exact solution, which may reach the required tolerance after a relatively small (compared to the problem size) number of iterations. The improvement is typically linear and its speed is determined by the condition number $\kappa(A)$ of the system matrix A: the larger is $\kappa(A)$, the slower the improvement.

4.5 The Discrete (Practical) Version

4.5.1 Step 1 (Fast Marching Algorithm for first equation)

Step 1 is usually performed using a fast marching algorithm for each sample. Two "active lists" are used, the A-list and the B-list. The A-list stores samples being processed, and the B-list stores samples to be processed after the current list is complete: [5]

⇒ For each sample:

⇒ Sample marked as 0 travel time.

⇒ Add sample to A-list

⇒ Until A-list is empty

- For each sample in A-list
  - Process sample
  - If converged
    - Add its neighbours to B-list if necessary
  - Else
    - Add sample itself to B-list
- Copy B-list to A-list.

When neighbours are checked, it only moves forward if the calculated traveltime for neighbours won't exceed the already calculated minimum.

4.5.2 Step 2 (Blockwise Conjugate Gradient Operations)

Block-wise conjugate gradient iterations are used in step 2, using blocks of 2×2.

$\Rightarrow \vec{q} \leftarrow \mathbf{A}\vec{x}$

$\Rightarrow \vec{r} \leftarrow \vec{b} - \vec{q}$

$\Rightarrow \vec{d} \leftarrow \vec{r}$

$\Rightarrow \delta \leftarrow \|\vec{r}\|^2$

$\Rightarrow \delta_r \leftarrow \sqrt{\delta}$

$\Rightarrow \delta_b \leftarrow \|\vec{b}\|$

$\Rightarrow \delta_s \leftarrow \epsilon\delta_b$

$\Rightarrow$ While $\delta_r > \delta_s$ and maximum iterations haven't been exceeded

- $\vec{q} \leftarrow \mathbf{A}\vec{d}$
- $dq \leftarrow \vec{d} \cdot \vec{q}$
- $\alpha \leftarrow \delta / dq$
- $\vec{x} \leftarrow \vec{x} + \alpha\vec{d}$
- $\vec{r} \leftarrow \vec{r} - \alpha\vec{q}$
- $\delta_{\text{old}} \leftarrow \delta$
- $\delta \leftarrow \|\vec{r}\|^2$
- $\beta \leftarrow \delta / \delta_{\text{old}}$
- $\vec{d} \leftarrow \vec{r} + \beta\vec{d}$
- $\delta_r = \sqrt{\delta}$

4.6 Chapter Summary

In this chapter the design and development stages of the project are explained in

detail, how the design of the algorithm include first the development of nearest neighbour interpolated image. Then time map is calculated which is used to blend the image to get the final blended output that will follow the original seismic image.

Chapter 5 ANALYSIS AND EVALUATION

5.1 Project Testing and Validation

This chapter demonstrates the performance analysis of algorithm developed for the project and also defines the project limitations.

5.2 Methodology/Work Flow

The algorithm starts off by computing the structure tensor $\mathbf{S}(\vec{x})$ for the given image or volume.

5.2.1 Computing the Structure Tensor

5.2.1.1 2-D Case

If we set$I_x(\vec{x}) = d/dx\,(I)$, and $I_y(\vec{x}) = d/dy\,(I)$, where I is the input guiding image then the structure tensor $\mathbf{S}(\vec{x})$ is defined as

$$\mathbf{S}(\vec{x}) = \begin{bmatrix} I_x^2(\vec{x}) & I_x I_y(\vec{x}) \\ I_x I_y(\vec{x}) & I_y^2(\vec{x}) \end{bmatrix}.$$

Practically, for noisy discrete signals, different difference operators are used, such as Sobel, Gaussian, Canny or Deriche. In our tests, we used the Gaussian operator with a horizontal radius of 7 and a vertical radius of 3.

5.2.1.2 3-D Case

The 3-D case is similar. We set $I_z(\vec{x}) = d/dz\,(I)$ and then

$$\mathbf{S}(\vec{x}) = \begin{bmatrix} I_x^2(\vec{x}) & I_x I_y(\vec{x}) & I_x I_z(\vec{x}) \\ I_x I_y(\vec{x}) & I_y^2(\vec{x}) & I_y I_z(\vec{x}) \\ I_x I_z(\vec{x}) & I_y I_z(\vec{x}) & I_z^2(\vec{x}) \end{bmatrix}.$$

The same considerations for practical computations also apply.

5.3 Computation of the Coherence and the Guiding Tensor

The coherence of the image can be calculated by the following formula at every point:

$$c(\vec{x}) = \frac{e_1(\vec{x}) - e_2(\vec{x})}{e_1(\vec{x}) + e_2(\vec{x})}$$

Where $e_1(\vec{x})$ is the maximum eigenvalue of $\mathbf{S}(\vec{x})$ at every point and $e_2(\vec{x})$ is the next largest. This is a sort of "dominance metric" which checks how much $e_1(\vec{x})$ dominates over $e_2(\vec{x})$, i.e., how strongly the layer is oriented in the given direction. If the difference is large, it is strongly oriented in one direction. If it is small, it's oriented weakly. This helps us detect faults, where the coherence is usually low.

The guiding tensor $\mathbf{D}(\vec{x})$ is computed as

$$\mathbf{D}(\vec{x}) = \frac{s\mathbf{S}^{-1}(\vec{x})}{1 - c(\vec{x})}.$$

This is the tensor used during the fast marching computations, with the scalar s normalizing the largest eigenvalue of the numerator to unity. During fast marching, this guiding tensor is used. However, during the conjugate gradient iterations, a different tensor is used:

$$\mathbf{D}(\vec{x}) = t^2(\vec{x})\vec{v}_1(\vec{x})\vec{v}_1^T(\vec{x})$$

Where $\vec{v}_1(\vec{x})$ is the eigenvector corresponding to the largest eigenvalue of $\mathbf{S}(\vec{x})$, and $t(\vec{x})$ is the traveltime computed during the fast marching process.

5.3.1 Fast Marching for Traveltime and Nearest Neighbour Computation

The fast marching method has already been adequately described in section 4.5.1, however, here we present its flowcharts.

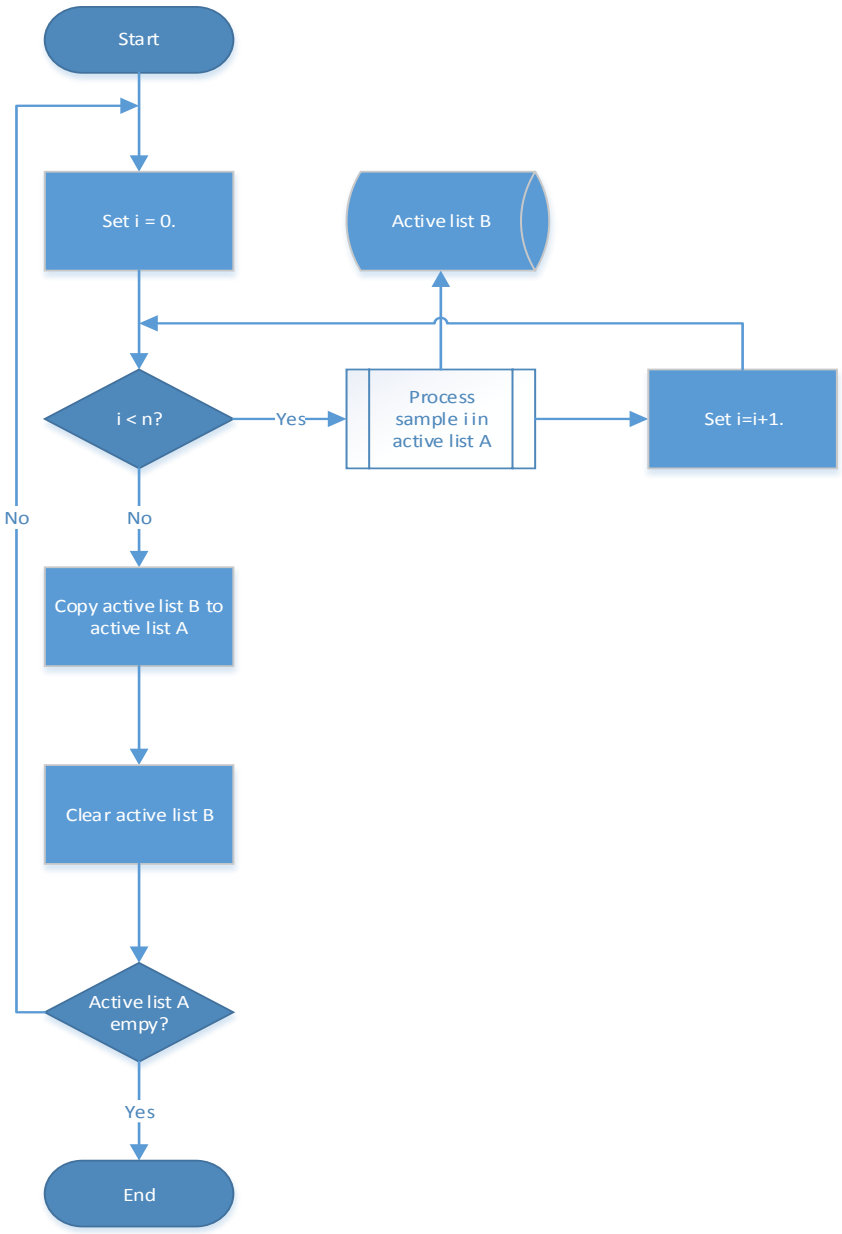


Figure 5-1: Processing a known sample.

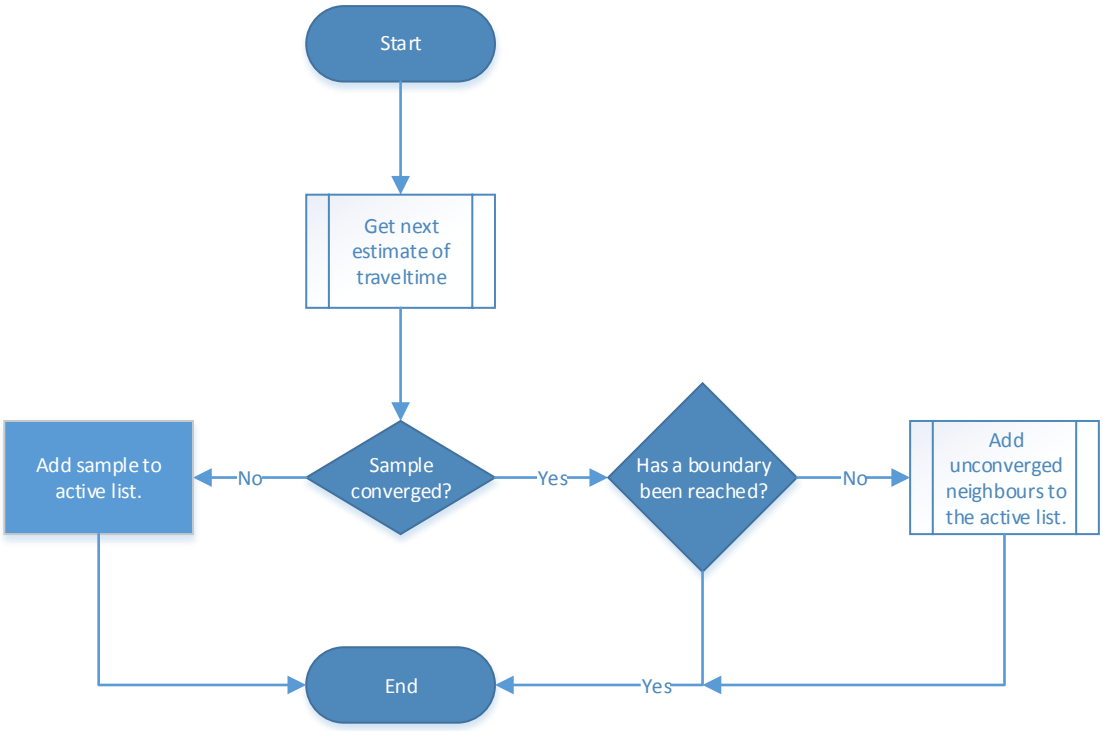


Figure 5-2: Processing a sample in an active list

5.3.2 Conjugate Gradient Iterations

Conjugate gradient iterations have also been covered in section 4.5.2, and as before, we present the flowchart here:

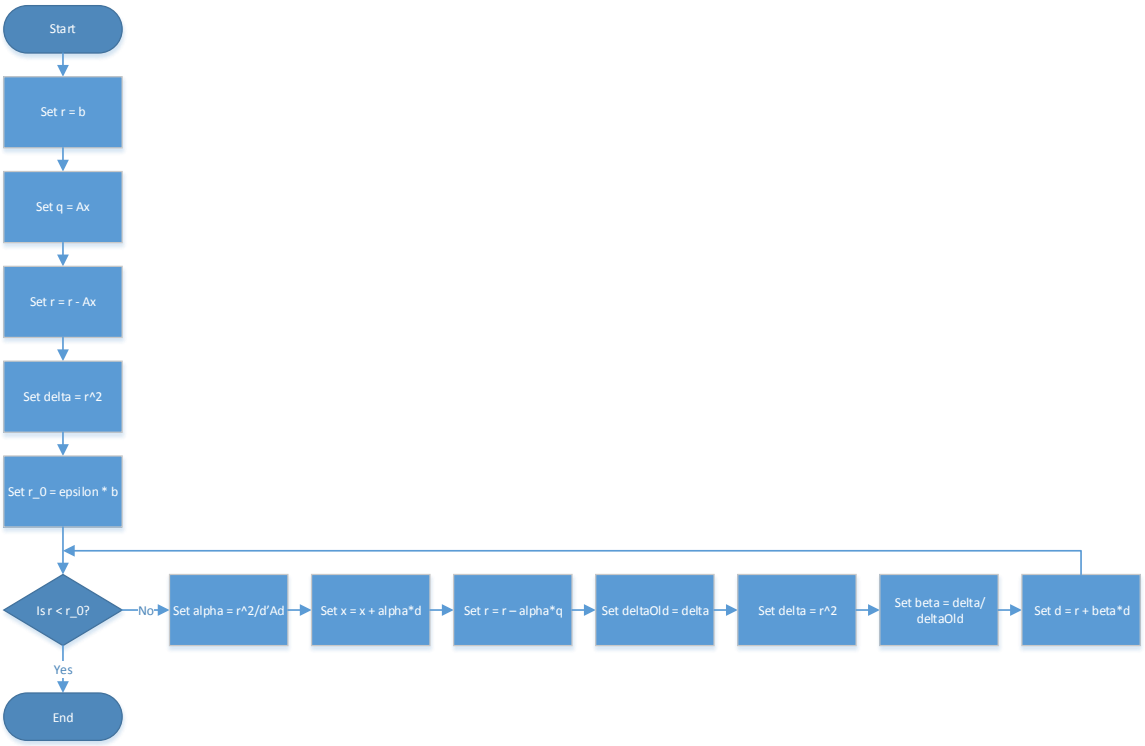


Figure 5-3: Conjugate-gradient iterations.

5.4 Results Explained

5.4.1 Input

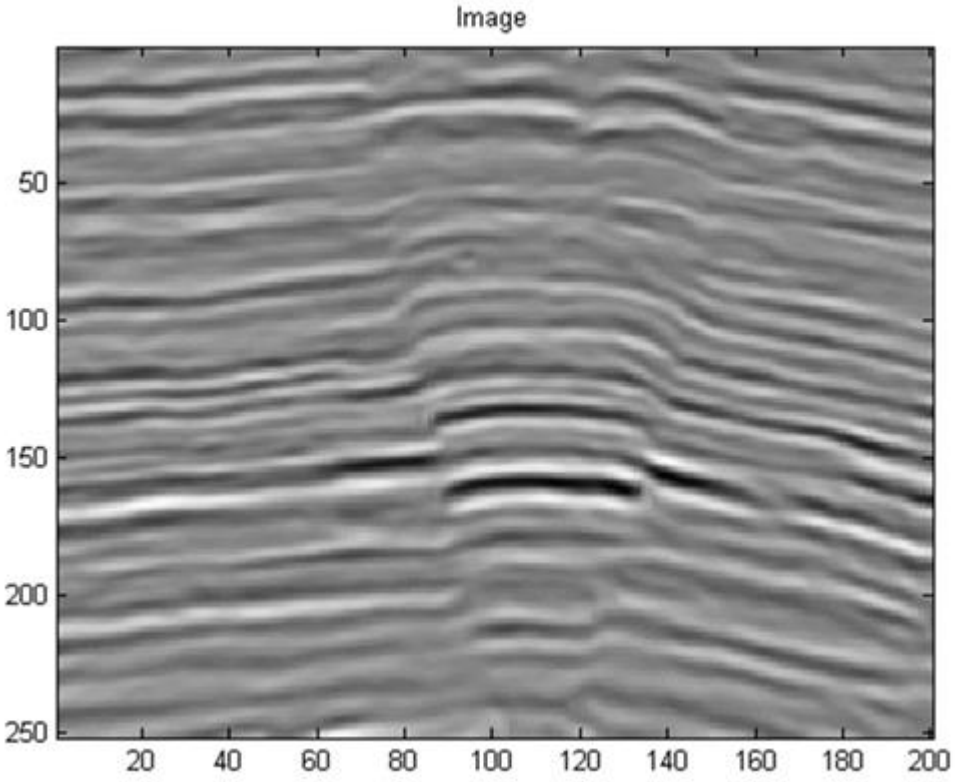


Figure 5-4: The input image.

5.4.2 Output after First Step

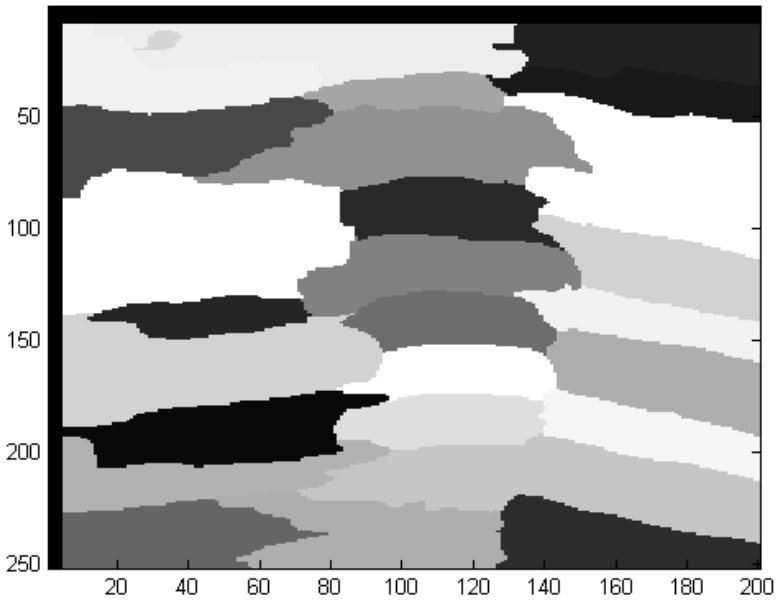


Figure 5-5: The nearest neighbour interpolant.

5.4.3 Time Map

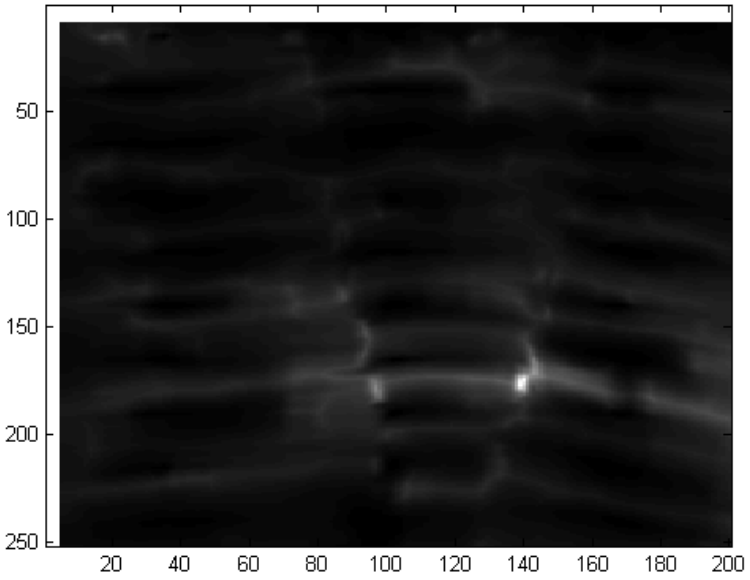


Figure 5-6: The traveltime.

5.4.4 Blended Near Neighbour Output

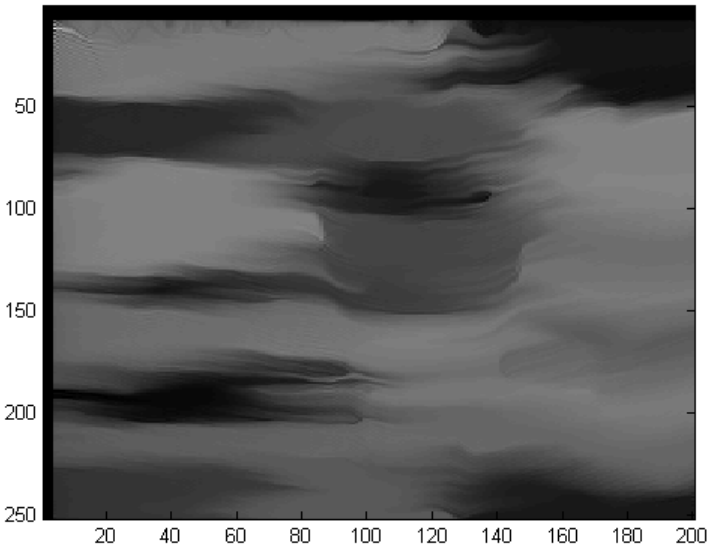


Figure 5-7: The blended output.

5.5 Limitations

The algorithm developed works perfectly fine with 3d volumetric data but for n-dimensional data it takes more processing time as compared to the 3d one.

5.6 Chapter Summary

This chapter deals with the testing and evaluation of the algorithm after a test bed had been completed in MATLAB. The results compiled had been field tested by LMKR and found to be over 95 percent accurate.

Chapter 6 FUTURE ENHANCEMENTS AND CONCLUSION

6.1 Development in this system

There could still be modifications made to this algorithm to make the algorithm even more accurate and time efficient. For example,

- The difference operator used for the calculation of the structure tensor could be changed to the Deriche operator for better results.
- The algorithm could be implemented on FPGAs for real-time processing.
- The mathematical representation of $\mathbf{D}(\vec{x})$ used in both steps could be tweaked for more accurate results.
- The coherence metric used was the simple eigenvalue metric… We could, instead, use other, more advanced forms of the coherence metric, such as that used in [3].

6.2 Conclusion

The Image guided blended near neighbour interpolation can be used for a variety of purposes. The main one being the efficient interpolation of well log data to seismic data.

Chapter 7 BIBLIOGRAPHY

[1] D. Hale, "Image-guided 3D interpolation of borehole data," Colorado School of Mines, Colorado, USA, 2010.

[2] D. Hale, "Image-guided blended neighbor interpolation of scattered data," Colorado School of Mines, Colorado, USA, 2009.

[3] D. Hale, "Structure-oriented smoothing and semblance," Colorado School of Mines, Colorado, USA, 2009.

[4] D. Hale, "Structure-oriented bilateral filtering," Colorado School of Mines, Colorado, USA, 2011.

[5] W.-K. Jeong and R. T. Whitaker, "A fast Eikonal equation solver for parallel systems," [Online]. Available: http://people.seas.harvard.edu/~wkjeong/publication/SIAM_CSE_07.pdf.

Anx-A

RECOMMENDATION FOR AWARD OF MEDAL

Mr. Hameer Abbasi is recommended/not recommended to be considered for award of medal based on his/her performance in his/her project titled

“Image Guided Blended Near Neighbor Interpolation and its Applications in Oil Exploration”.

_____

Brig (R) Muhammad Khan Minhas

EE Department

MCS