QUANTIFICATION OF POROUS MEDIA USING GAMMA RAY TOMOGRAPHY

By

HSUAN-TSUNG HSIEH

Bachelor of Science
National Taiwan University
Taipei, Taiwan, R. O. C.
1986

Master of Science
University of Rochester
Rochester, New York
1992

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 1997

OKLAHOMA STATE UNIVERSITY

QUANTIFICATION OF POROUS MEDIA

USING GAMMA RAY TOMOGRAPHY

Thesis Approved:

Glenn Brown
Thesis Adviser

Douglas C. Kent

Wayne B Powell
Dean of the Graduate College

ACKNOWLEDGMENTS

I wish to express my sincere appreciation to my major advisor, Dr. Glenn O. Brown for his intelligent supervision, inspiration and friendship throughout this research. My honest appreciation extends to my other committee members, Dr. M. L. Stone, Dr. C. T. Haan, Dr. D. Kent and Dr. A. W. Hounslow, for their valuable assistance. I would like to thank the Department of Biosystems and Agricultural Engineering for providing the laboratory facilities which made this study possible.

Financial support was provided in part by Sandia National Laboratories, Albuquerque, New Mexico for the United States Department of Energy under contract DE-AC04-94-AL8500. Dan A. Lucero was the Sandia technical representative. Chapter 2 had undergone Sandia technical review.

My deepest appreciation goes out to my grandmother, Kou-Chou Hsu, who could not live to see my completion. I could not finish this thesis without my sister and brothers, Kai-lin, Shen-Tsung and Lang-Tsung. For their strong supporting, understanding and sacrifice at times of difficulty, any appreciation would be too trivial. Thanks also go to my parents, Su-Mei Hsu and Jy Hsieh, for their support and encouragement throughout my life.

Finally, I would like to give my deepest thanks to my wife and my best friend, Meihua Koo, for her unconditional, optimistic and faithful support during this research and throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure		**Page**

Figure **Page**

Figure **Page**

Chapter I

BACKGROUND AND OBJECTIVES

Background

The nature of the interior structures within porous media that control transport phenomena has been puzzling researchers for decades. Similar samples operated under identical flow conditions can yield completely different leaching breakthrough curves and have permeability several orders of magnitude different. To try and explain these differences we would like to know,

- the individual pore size and the distribution,
- the arrangement and interconnection of pores,
- the distribution of water-filled pores and their involvement in the water and solute transport processes, and
- how to discern different rates of water and solute transport through pores.

Researchers have applied varied technologies, such as conceptualized models, thin-section impregnation, column leaching and permeameters, to define those parameters in laboratory-scale samples. As examples, *Long and Witherspoon* [1985] and *Tsang and Tsang* [1987] used numerical modeling to illustrate the correlation between permeability in heterogeneous porous media and the fracture geometry while *Bouma et al.* [1977] and

Singh et al. [1991] applied thin-sectioning and impregnation techniques to quantify macropores and fractures properties of soil. However, most conventional methods measure physical properties at a core-averaged scale and many times at the cost of sample destruction. Thus a search for better techniques to quantify laboratory-scale samples with non-destructive and noninvasive natures becomes important. Computerized Tomography (CT) provides such capability for studying porous media behavior in space and time. It has been increasingly utilized in soil and ground water research for the past decade. While non-destructive, it provides the same or a better understanding of porous structures as thin sectioning. Several notable applications are found in *Anderson and Hopmans* [1994]. Those studies and others have examined the potential for tomographic measurements in quantifying various porous media properties, including porosity, bulk density, soil water content, and transport processes [*Petrovic et al.*, 1982; *Hainsworth and Aylmore*, 1983; *Crestana et al.*, 1985]. Tomographic image resolution ranges from a few cubic millimeters for gamma or conventional X-ray CT, to as small as a few cubic micrometers for synchrotron CT [*Spanne et al.*, 1994].

Unfortunately, though CT can provide a qualitative representation of structures, the lack of systematic and theory-based image analysis algorithms have limited in-depth image interpretation. Several researchers have made efforts into quantification of macropores and fractures [*Anderson et al.*, 1990; *Hopmans et al.*, 1994; *Warner et al.*, 1989; *Warner et al.*, 1991; *Greves et al.*, 1989]. *Peyton et al.* [1992] developed an iterative procedure that considers macropore spatial relationship and evaluates variously sized macropores. *Kantzas* [1991] has compared the bulk density and effective atomic number to discriminate sulfur within a dolomitic rock. However, most quantified results that relate component contents and feature properties in CT imaging are determined using simple threshold methods and attenuation frequency distribution curves [*Spanne et al.*, 1994; *Kantzas*, 1990; *Kantzas et al.*, 1991]. While applying ray CT as a quantification tool, no one has extensively analyzed the attenuation frequency distributions and their correlation to the associated photon statistical errors, component content and heterogeneity.

Elementary Tomography

To improve the understanding of CT images, the basic radiation phenomena and elementary tomography have to be considered. *Brown et al.* [1993] provided a comprehensive theoretical review and system configuration of tomographic measurements using transmission radiation and developed the CT system used here. Figure 1-1 shows the gamma ray CT system. Target objects are placed on a movable stage between the lead shielded source and a 2-in NaI (T1) detector. A 50 mm long replaceable tungsten collimator is used in front of both source and detector. The replaceable collimators provide 1 to 5 mm collimation. Collimation must be in good alignment, in order for gamma-rays to pass from the source through the target sample and into the detector with minimum energy loss. Adjustment can be done by adjusting the detector stage to allow a laser to shine through the detector collimation into the source collimator. One rotary and two linear tables are directed by three computer-controlled stepper motors with horizontal and vertical position repeatability of 0.005 mm and the rotary precision of 0.5 arc min. The detector signal is processed by a Ortec 925

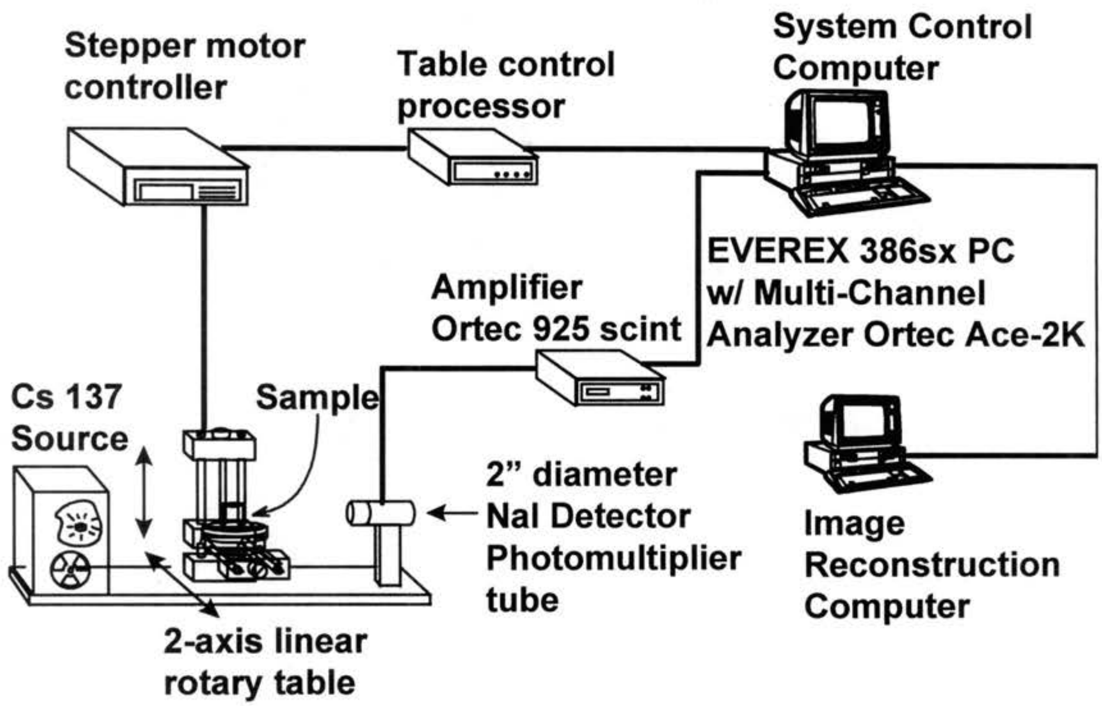


Fig. 1-1.Schematic of gamma ray tomography system, developed by *Brown et al.* [1993].

scintillation amplifier and Ace 2000 multi-channel analyzer. The personal computer-based analyzer allows for the automatic compensation of detector drift, window calibration, and data storage.

Figure 1-2 presents the scanning geometry using a single detector and parallel pencil beam source. The basic concept of CT scanning is to measure photon attenuation through the sample at various positions and angles so that all points in the scanning domain have been covered. That data can be restored to linear attenuation coefficients for each point within the domain by the appropriate reconstruction algorithm. More details can be found in *Kak and Slaney* [1988]. The exponential decay relationship between the initial source count I_0 and the attenuated count at the detector, I, over a constant time is:

$$I = I_0 \, exp(-\int_{\Delta L} \mu dL) \tag{1-1}$$

where μ is the voxel attenuation coefficient and L is the ray path. Therefore, each projection line integral, p, can be approximated as a summation of the voxel attenuation coefficients along the scanning path. The attenuation of the beam is given by

$$p(r,\phi) = ln[I_0 / I(r,\phi)] = \sum_{L(r,\phi)} \mu(x,y)\Delta L \tag{1-2}$$

where ΔL is the voxel path length, r is the ray position and ϕ is the projection angle. In this study, the convolution back-projection function of *Shepp and Logan* [1974] is used to reconstruct the attenuation image. A discrete approximation of this continuous function at M discrete projection angles and N discrete positions is given by

$$\mu(x,y) = \Delta\phi\tau \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} p(n\tau, m\Delta\phi)\gamma(x\cos m\Delta\phi + y\sin m\Delta\phi - n\tau) \tag{1-3}$$

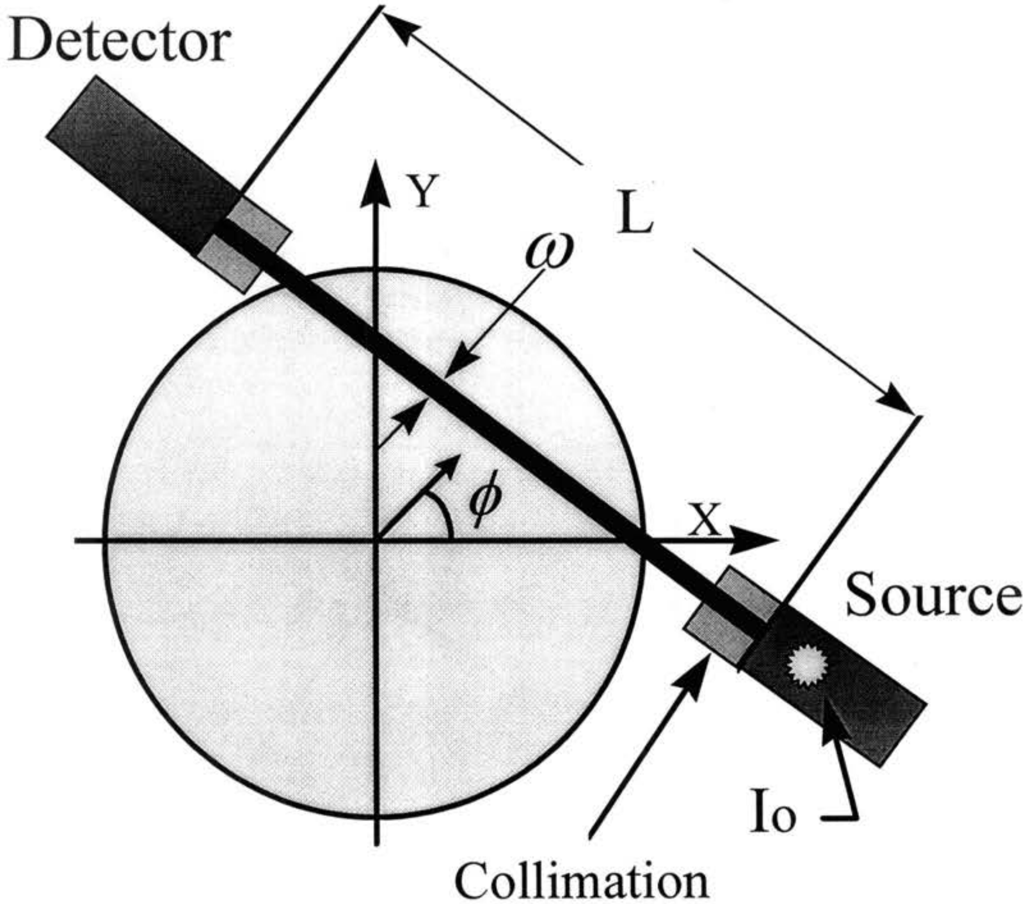


Fig. 1-2. Geometry of parallel pencil beam tomography.

where $\Delta\phi$ is the angular step between projections, τ is the distance between rays, m and n are the summation counters, and γ is a filter function. As shown in *Gardner* [1986] the attenuation coefficient of each voxel is directly related to the density and atomic number of the material within the voxel region,

$$\mu(x,y) = f(\rho(x,y), z(x,y)) \tag{1-4}$$

where ρ is density and z is the atomic number. The atomic number dependency is small for most geologic materials. So to a reasonable accuracy,

$$\mu(x,y) = C\rho(x,y) \tag{1-5}$$

where C is a calibration factor. Image analysis can be performed by either attenuation coefficient or calibrated density. While using CT scanning, photon counts from radiation-source emissions have a complicate Poisson distribution that can be approximated by a normal distribution for large counts. Therefore, each measured volume-element (voxel) attenuation value is also impacted by photon statistical noise. Since generating the attenuation coefficient image requires application of a complicate mathematical procedure, cooperating both measurement theory and image processing technique becomes a major barrier for advanced data analysis.

Objectives

Qualitative descriptions of CT, while straightforward and intuitive, provide limited quantitative measures that may be integrated with existing porous media theory. Since the lack of theory-supported algorithms is the leading restriction for the progress of CT applications, the overall objective of this study is to develop and demonstrate two improved data analysis algorithms that step from qualitative image description to theory-supported quantitative data acquisition. The three specific tasks to meet this objective are:

1. Determine the relationship between the pure component frequency distribution and the degree of heterogeneity in a sample, and develop a general procedure to analyze the CT attenuation frequency distribution,
2. Develop a comprehensive, semi-automatic CT image segregation algorithm that combines fundamental CT theory with data characteristics, and
3. Demonstrate the proposed algorithms by exploring the fundamental concept of the representative elementary volume (REV) using a real sample.

Chapter II addresses the first task and shows how porous media compositions can be quantified by applying a statistical deconvolution approach. The same chapter also explores the relationship between the density frequency distribution and the component volume content and heterogeneity using a complex sample core. Chapter III presents a theory-based component segregation algorithm that can accurately distinguish structural features and pure component distributions in CT images. Such new threshold-independent algorithm eliminates the disturbance caused by arbitrary threshold selection used by most researchers in the past for component identification. Chapter IV applies those algorithms to generate results that describe pure component and macroporosity distributions. Two samples of the Culebra Dolomite Member of the Rustler Formation collected at the Waste Isolation Pilot Plant near Carlsbad, New Mexico, are used to perform the procedure. Besides property identification, the results provide a new insight into a problem that have plagued porous media research [*Bear*, 1972; *Baveye and Sposito*, 1984], the identification and size of the REV. This initiates the attempt to bridge CT scanning to porous media theory. Finally, Chapter V summaries and provides future research recommendations.

References

Anderson, S. H., R. L. Peyton and C. J. Gantzer, Evaluation of constructed and natural soil macropores using x-ray computed tomography. *Geoderma*, 46, 13-29, 1990.

Anderson, S. H., and J. W. Hopmans (Ed.), *Tomography of Soil-Water-Root Processes,* Soil Sci. Soc. Am. Special Publication No. 36, Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Baveye, P. and G. Sposito, The operation significance of the continuum hypothesis in the theory of water movement through soils and aquifers, *Water Resour. Res.*, 20, 521-530, 1984.

Bear, J., *Dynamics of Fluids in Porous Media*, American Elsevier, New York, 1972.

Bouma, J, A. Jongerius, O. Boersma, A. Jager and D. Schoonderbeek, The function of different types of macropores during saturated flow through four swelling soil horizons. *Soil Sci. Soc. Am. J.*, 41, 945-950, 1977.

Brown, G. O., M. L. Stone and J. M. Gazin, Accuracy of gamma ray computerized tomography in porous media, *Water Resour. Res.*, 29(2), 479-486, 1993.

Crestana, S., S. Mascarenhas, and R. S. Pozzi-Mucelli, Static and dynamic three-dimensional studies of water in soil using computed tomographic scanning, *Soil Sci.*, 140(5), 326-332, 1985.

Gardner, W., Water content, in *Methods of Soil Analysis, Part 1 Physical and Mineralogical Methods* - Agronomy Monograph no. 9, 2nd ed., Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, pp. 493-544, 1986.

Greves, M.C.J., De Jong and R.J.St. Arnaud, The characterization of soil macroporosity with CT scanning, *Can. J. Soil Sci.*, 69, 629-637, 1989.

Hainsworth, J. M., and L. A. G. Aylmore, The use of computer-assisted tomography to determine spatial distribution of soil water content, *Aust. J. Soil Res.*, 21(4), 435-443, 1983.

Hopmans, Jan W., Milena Cislerova, and Tomes Vogel. 1994. X-ray tomography of soil properties, in *Tomography of Soil-Water-Root Processes*, edited by S. H. Anderson and J. W. Hopmans, pp. 17-28, SSSA Special Publication No. 36. Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Kak, A. C., and M. Slaney, *Principles of Computerized Tomographic Imaging,* pp. 8-9, pp. 194-197, IEEE Press, New York, 1988.

Kantzas, A., Investigation of physical properties of porous rocks and fluid flow phenomena in porous media using computer assisted tomography, *In Situ*, 14(1), 77-132, 1990.

Kantzas, A., Determination of sulfur saturation dolomitic sour gas reservoir using computer assisted tomography, *In Situ,* 15(3), 215-246, 1991.

Long, J.C.S. and P.A. Witherspoon, The relationship of the degree of interconnection to permeability in fracture networks, *J. Geophys. Res.*, 90(B4), 3087-3098, 1985.

Petrovic, A. M., J. E. Siebert, and P. E. Rieke, Soil bulk density analysis in three dimensions by computed tomographic scanning, *Soil Sci. Soc. Am. J.*, 46(3), 445-450, 1982.

Peyton, R.L., B.A. Haeffner, S.H. Anderson and C.J. Gantzer, Applying x-ray CT to measure macropore diameters in undisturbed soil cores. *Geoderma*, 53, 329-340, 1992.

Shepp, L. A., and B. F. Logan, The fourier reconstruction of a head section, *IEEE Trans. Nucl. Sci.*, 21(3), 21-43, 1974.

Singh, P., R.S. Kanwar and M.L. Thompson, Measurement and characterization of macropores by using AUTOCAD and Automatic Image Analysis, *J. Environ. Qual.*, 20, 289-294, 1991.

Spanne, P., K. W. Jones, L. Prunty, and S. H. Anderson, Potential applications of synchrotron computed microtomography to soil science, in *Tomography of Soil-Water-Root Processes*, edited by S. H. Anderson and J. W. Hopmans, pp. 43-58, SSSA Special Publication No. 36. Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Tsang, Y.W. and C.F. Tsang, Channel model of flow through fractured media, *Water Resour. Res.*, 23(3), 467-479, 1987.

Warner, G. S., J. L. Nieber, I. D. Moore and R. A. Geise, Characterizing macropores in soil by computed tomography, *Soil Sci. Soc. Am. J.*, 53(3), 653-660, 1989.

Warner, G.S. and J.L. Nieber, Macropore distribution in tilled vs. grass-surfaced cores as determined by computed tomography, *Preferential Flow,* 192-201, 1991.

Chapter II

MEASUREMENT OF POROUS MEDIA COMPONENT CONTENT AND HETEROGENEITY USING GAMMA RAY TOMOGRAPHY

Abstract

Tomographic images of porous media are complex distributions of linear attenuation coefficients, which reflect the combined effects of scanning spatial resolution, photon statistical measurement errors, and true material densities. I address how the true voxel-scale attenuation distribution and measurement errors are convoluted to yield measured density frequency distributions. A deconvolution algorithm is demonstrated that uses the measured density frequency distributions and known photon statistical errors to quantify average cross-section volume contents of pure components and a mixed-component phase. The mixed-component phase represents regions where components are intertwined or varied in spaces smaller than the scanning resolution. This approach is applied to a complex core of the Culebra Dolomite Member of the Rustler Formation collected at the Waste Isolation Pilot Plant, near Carlsbad, New Mexico. The methodology provides a quantitative measure of the volume content of gypsum, dolomite and mixed-components, and heterogeneity in the sample.

Introduction

Computerized tomography (CT) is being utilized increasingly in soil and ground water research. Several notable applications are found in *Anderson and Hopmans* [1994]. Those studies and others have looked at the potential for tomographic measurements to quantify various porous media properties, including porosity, bulk density, soil-water content, and transport processes [*Petrovic et al.*, 1982; *Hainsworth and Aylmore*, 1983; *Crestana et al.*, 1985]. Tomographic spatial resolution ranges from a few cubic millimeters for gamma or conventional x-ray CT, to as small as a few cubic micrometers for synchrotron CT [*Spanne et al.*, 1994]. One fundamental property that may be obtained from tomographic measurements is the small-scale distribution of bulk density, which can provide information on sample composition and transport properties.

All radiation-source emissions are random events. The measured photon count will follow a Poisson distribution that can be approximated by a normal distribution for large counts. This photon statistical error will produce normally distributed noise in the volume-element (voxel) attenuation values. In the past, most quantitative measurements of soil and rock samples using CT scanning have not rigorously combined CT theory and data characteristics. Different sample compositions have been visualized on a density frequency distribution [*Spanne et al.*, 1994; *Kantzas,* 1990; *Kantzas et al.*, 1992], and the voxel gray-scale distribution has been used to quantify components and structures in CT images by arbitrarily selecting thresholds [*Hopmans et al.*, 1994; *Kantzas*, 1990; *Peyton et al.*, 1994]. However, no one has analyzed the density frequency distribution with its associated photon statistical errors to yield quantitative estimates of component volume content and heterogeneity in the sample.

This research demonstrates how porous media compositions can be quantified by applying a statistical approach to tomographic measurements obtained from gamma-ray computerized tomography (Gamma CT). What is more important, the relationship between the density frequency distribution and the component volume content and heterogeneity in a sample can be determined. These procedures are general in nature and may also be applied to x-ray and synchrotron images, subject to the limitations of the specific machine and porous media system.

Tomography Measurements

Measurement Theory

Theoretical foundations that define tomographic measurements using transmission radiation are briefly developed below. *Brown et al.* [1993] provide a more comprehensive review. CT images are generated by measuring photon attenuation on many different paths through a sample. A relationship exists between the number of incident photons, I_0, on the source side of the target with that on the detector side, $I(r, \phi)$, where r is the ray position and ϕ is the projection angle. This relationship is the line integral of the attenuation coefficient of the material, along the beam path of length L and may be approximated as a summation of the voxel attenuation coefficients, $\mu(x, y)$, along the path. The attenuation coefficient is directly related to the bulk density of rock mass and the container material along the pathway [*Gardner,* 1986]. The attenuation of the beam with initial count I_0 and attenuated count $I(r, \phi)$ over a constant time is given by

$$p(r,\phi) = ln[I_0 / I(r,\phi)] = \sum_{L(r,\phi)} \mu(x,y)\Delta L \qquad (2\text{-}1)$$

where p is the projection line integral and ΔL is the voxel path length. The beam intensities may be in units of either counts per time or total counts for a constant counting time. The convolution back-projection function of *Shepp and Logan* [1974] is traditionally used to obtain the attenuation image. A discrete approximation of this continuous function at M discrete projection angles and N discrete positions is given by

$$\mu(x,y) = \Delta\phi\tau \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} p(n\tau, m\Delta\phi)\gamma(x\cos m\Delta\phi + y\sin m\Delta\phi - n\tau) \qquad (2\text{-}2)$$

where $\Delta\phi$ is the angular step between projections, τ is the distance between rays, m and n are the summation counters, and γ is a filter function. The major source of errors in CT systems is photon statistical errors, arising from the random nature of photon emissions. For large photon count, any given count, I, will follow a normal distribution with a mean and variance of I^*. If the true value of I_0 is known, *Kak and Slaney* [1988] showed that the variance image of a reconstruction is normally distributed with

$$\sigma^2(\mu(x,y)) = (\Delta\phi\tau)^2 \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \frac{\gamma^2(x\cos m\Delta\phi + y\sin m\Delta\phi - n\tau)}{I^*(n\tau, m\Delta\phi)} \qquad (2\text{-}3)$$

where $\sigma^2(\mu(x, y))$ is the variance of the voxel attenuation. If I is used as an estimate of I^*, Eq. 2-3 allows the computation of a photon variance image corresponding to the object image, given by Eq 2-2.

Petrovic et al. [1982] and *Orsi et al.* [1994] found that for x-ray CT, attenuation was a linear function of bulk density in the single mineral soil and rock samples they used but was influenced by mineral composition. *Luo and Wells* [1992] showed that mass attenuation coefficients are insensitive to mineral composition at the 662 keV gamma-ray energy used here. They demonstrated that for nine very different soils the theoretical attenuation coefficient varied only 1%. Thus, bulk density and attenuation are linearly related in Gamma CT images. For ease of discussion they may be considered equivalent, and related by a system calibration factor. Normally systems are calibrated with a single mineral standard of known density. This calibration factor, C may be defined as

$$C = \frac{\rho_p}{\mu_c} \tag{2-4}$$

where ρ_p is the mineral standard's density and μ_c is the attenuation coefficient. It follows that the bulk density, ρ and its variance, σ^2, are computed by

$$\rho(x, y) = C\,\mu(x, y) \tag{2-5}$$

and

$$\sigma^2(\rho(x, y)) = C^2 \sigma^2(\mu(x, y)) \tag{2-6}$$

Calibration is performed by scanning a mineral sample and reconstructing its attenuation and variance images. Using vision-analysis software, the image is examined and a large area of pure mineral away from any boundaries is located. The attenuation of the mineral is given by the mean of the area's voxel attenuation values. An assurance that only pure mineral is present and the machine is operating properly is given by computation of the voxel attenuation variance inside the area, and comparison to the variance image value. The two will only be equal if the mineral is pure.

Origin of Measured Density Distribution

Consider a sample with only a single uniform component with a true density of ρ^*. A true voxel density frequency distribution can be defined as a Dirac delta function at ρ^* with zero value elsewhere, as shown in Figure 2-1 and expressed as

$$Q(\rho) = 0 \qquad \rho \neq \rho^* \tag{2-7}$$

with

$$\int_{-\infty}^{\infty} Q(\rho) d\rho = 1 \tag{2-8}$$

where $Q(\rho)$ is the density frequency distribution. Photon statistical errors transform the true distribution by the Gaussian distribution, $g(\rho)$:

$$g(\rho) = \frac{1}{\sqrt{2\pi\sigma^2}} exp[-\frac{(\rho - \rho^*)^2}{2\sigma^2}] \tag{2-9}$$

The approximated distribution $f(\rho)$ is the convolution of the Dirac delta with the Gaussian [*Kak and Slaney*, 1988]:

$$f(\rho) = Q(\rho) * g(\rho) \qquad -\infty < \rho < \infty \tag{2-10}$$

where $*$ indicates the convolution of Q and g. This convolution process is shown in Figure 2-1.

Two complications can be expected. First, rocks typically have more than one component, including multiple minerals and pore space. Second, when two or more components are interlaced with one another, a large number of voxels has component boundaries crossing through them. Those voxels will have a true density-between the density of the pure components. The true density, ρ_m^* of a "mixed-component" voxel at position (x, y) made up of K components will be given by

$$\rho_m^*(x, y) = \sum_{k=1}^{K} r_k(x, y) \rho_k^* \tag{2-11}$$

where $r_k(x, y)$ is the dimensionless volume content of the component k in the individual voxel, ρ_k^* is the true pure component attenuation coefficient, and K is the total number of

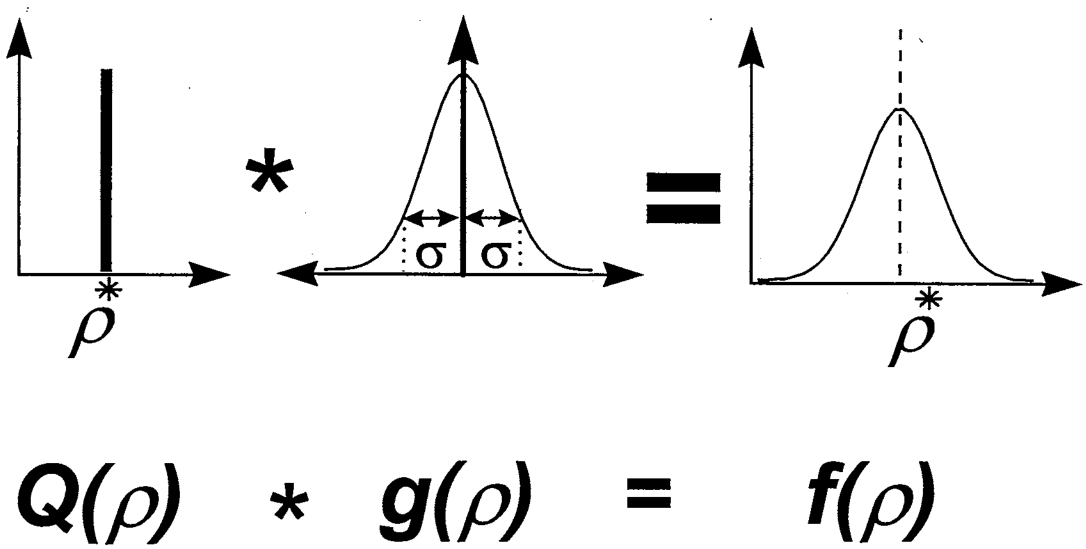


Figure 2-1. Convolution of a Dirac delta function with a Gaussian distribution.

components in the voxel. Eq. 2-11 neglects partial-volume effects that result from the geometric alignment of boundaries within a voxel [*Gardner*, 1986]. Those effects can be shown to be trivial when $\Delta L << 1/\mu$. Since boundaries can fall anywhere in a voxel, the expected distribution for the mixed voxels would not be a Dirac delta function, but instead a relatively uniform distribution spanning the density range between the two components.

The complexity of the mixed-component density frequency distribution can be expected to increase with the number of components in the sample, while its magnitude will increase with phase mixing at the voxel length scale. As more components are blended, a larger number of mixed-component voxels will occur. The mixed-component voxel frequency distribution over the range $0 \le \rho_m \le \rho_{max}$ is reasonably approximated by the beta distribution with parameters of α and β, given by

$$f_m^*\left(\frac{\rho_m}{\rho_{\max}}\right) = \begin{cases} \left[\left(\frac{\rho_m}{\rho_{\max}}\right)^{\alpha-1}\left(1-\frac{\rho_m}{\rho_{\max}}\right)^{\beta-1}\right]\frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} & 0 \le \frac{\rho_m}{\rho_{\max}} \le 1 \\ 0 & elsewhere \end{cases} \tag{2-12}$$

where $f_m^*(\rho_m/\rho_{max})$ is the true mixed-component phase distribution, Γ is the gamma function, and ρ_{max} is defined as the largest true density in a sample. Since bounded, positive functions with arbitrary distribution curves are expected in the mixed-component phase, the beta distribution with a similar characteristic [*Port*, 1994] is preferred. This distribution is recommended by *Mendenhall et al.* [1990] for describing the proportion of impurities in a chemical product and has a convenient property of reducing to a uniform distribution if $\alpha = \beta = 1$.

For a sample with a large volume of multiple pure components and a mixed-component phase, the true density frequency distribution, $f^*(\rho)$, will be given by the weighted sum of Eq.2-7, 2-8, and 2-12:

$$f^*(\rho) = \sum_{k=1}^{K} R_k(\rho)Q_k(\rho) + R_m(\rho)f_m^*(\frac{\rho}{\rho_{max}}) \quad (2\text{-}13)$$

where R_k is the fractional volume content of the pure component k, and R_m is the volume content of the mixed phase. It follows that $R_k, R_m \geq 0$ and

$$(\sum_{k=1}^{K} R_k) + R_m = 1 \quad (2\text{-}14)$$

Figure 2-2 shows $f^*(\rho)$ for a three-phase system. As depicted it would represent a sample with two minerals and empty pores larger than the voxel size. The approximated frequency distribution for the sample, $f(\rho)$, is then given by the convolution of the true density distribution in Eq. 2-13 with the Gaussian:

$$f(\rho) = f^*(\rho) * g(\rho) \quad (2\text{-}15)$$

As shown in Figure 2-2, the various components produce a continuous mixed-component phase distribution with broad peaks of pure components. The beta distribution is only slightly modified by the Gaussian convolution. This results from overestimates at a given density being balanced by underestimates at a higher density. Significant tailing of the beta distribution occurs only at the steep end of the distribution. Other samples could show greatly different distributions, and the mixed component could represent a greater fraction of the porous media.

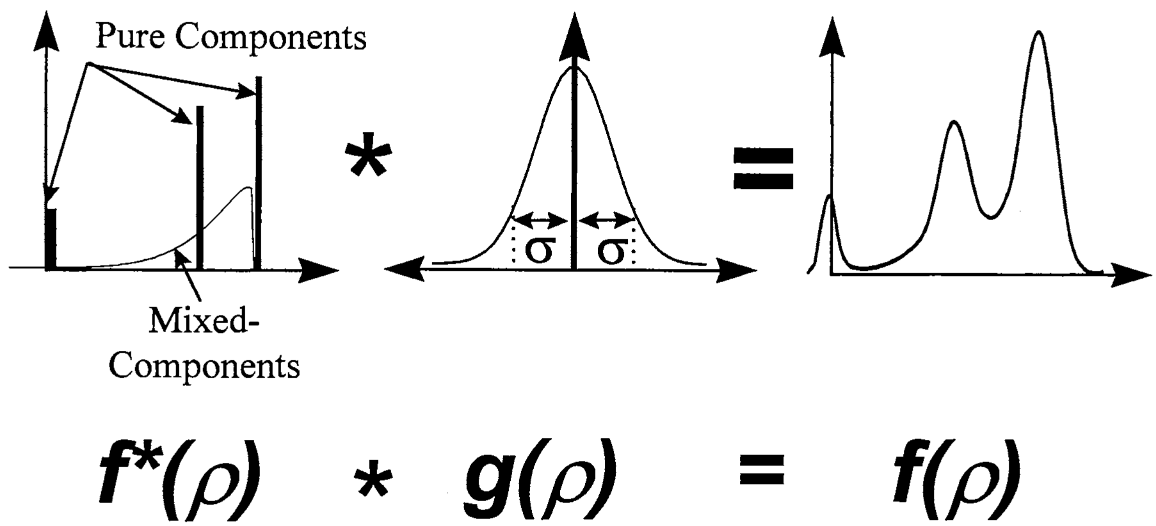


Figure 2-2. Convolution of multiple Dirac delta functions and a mixed voxel distribution with a Gaussian distribution.

Parameter Identification

The measured density distribution represents a summation of pure components, a mixed-component phase and photon statistical errors, given by Eq. 2-15. Thus, component content can be quantified by fitting the parameters in Eq. 2-12 and 2-13 to measured data. In addition to σ^2 calculated by Eq. 2-3, each pure component will require defining two parameters, R_k, and ρ_k^*, while the mixed-component phase will require four, R_m, α, β, and ρ_{max}. For example, a set of 10 parameters will be required for fitting a three-phase system. However, since the sum of the component contents must equal one and ρ_{max} is defined as the maximum end-component attenuation which is equal to the largest ρ_k^*, only eight will be independent. If each pure component's density can be measured or estimated independently, the problem will be reduced to fitting five parameters

Fitting five parameters can be problematic, however, assuming the measured density frequency distributions have distinctive regions where only one of the parameters is significant, a sequential, trial and error parameter fitting procedure may be used. First, the continuous density frequency distribution in Eq. 2-15, is converted to a discrete relative distribution, $F(\rho)$ by

$$F(\rho) = \Delta\rho \, f(\rho) \tag{2-16}$$

where $\Delta\rho$ is the class interval for the distribution. Next, each peak in the measured relative density frequency distribution is postulated as a pure component at that density and used in the Dirac delta function of Eq. 2-7 and 2-8. An average attenuation variance over the region of interest is calculated from Eq. 2-3 and 2-6. Then α and β are fitted to match the general shape of the distribution in the region from zero to ρ_{max}. The standardized residual error, defined as the ratio of the difference between the measured and estimated values to the measured value, is used. Finally, the volume fraction of pure components and mixed-component phase, R_k and R_m are estimated by iterative adjustments to minimize the residuals to the second decimal point between the measured and fitted distributions.

The *chi-square* goodness of fit, χ_c^2, may be used to quantify the adequacy of a fitted distribution to observed data [*Haan,* 1977] and is given by

$$\chi_c^2 = \sum_{j=1}^{J} \frac{[F_o(\rho_j) - F_E(\rho_j)]^2}{F_E(\rho_j)} \tag{2-17}$$

where j is the density frequency interval, J is the total number of class intervals, $F_E(\rho_j)$ is the estimated absolute density frequency, and $F_o(\rho_j)$ is the measured frequency. The hypothesis that the data are from the fitted distribution is rejected if

$$\chi_c^2 > \chi^2_{(1-v,J-i-1)} \tag{2-18}$$

where $\chi^2_{(1-v,\ J-i-1)}$ is the critical value of *chi-square* with i estimated parameters, $(1-v)$ confidence level, and $(J-i-1)$ degrees of freedom.

Materials and Methods

Sample

A dolomite core, 145 mm in diameter and 100 mm long, from the Culebra Dolomite member of the Rustler Formation was used in this study [*Lucero et al.*, 1994]. It was collected by horizontal drilling at a depth of 218 m in the air intake shaft of the Waste Isolation Pilot Plant located near Carlsbad, New Mexico. Examination by scanning electron microscope of a separate sample from the same level and location found dolomite and gypsum, and trace amounts of corrensite, quartz, and halite. Visual examination of the core confirmed that dolomite and gypsum were the only significant mineral components. Significant gypsum infilling and dissolution channels were visible at the core ends shown in the photograph Figure 2-3a, and large voids are known to occur in other samples of the same material. The core was intended for column leach testing, thus it had a poured rubber lining, which filled any voids on the outer perimeter, but did not intrude deeply. Because of concerns with leaching test conditions, the core could not be dried or saturated before scanning, thus all measurements were made on a freely drained, moist sample. It is assumed that the secondary porosity was drained, but the primary porosity remained saturated. *Kelley and Saulnier* [1990] performed extensive measurements on 25, 50-mm Culebra core samples, and found the median dolomite grain density to be 2.83 Mg/m^3 with an average total porosity, including fractures of 0.13. Assuming a dolomite water filled primary porosity of 0.11, yields a dolomite wet bulk density of 2.63 Mg/m^3.

CT Scanning

The custom pencil-beam, gamma-ray CT scanner of *Brown et al.* [1993] was used. The monochromatic nature of gamma-ray transmission combined with monochromatic detection techniques allows voxel-scale densities to be determined with bounded error. Use of gamma rays eliminates beam-hardening phenomena and photon scatters common to x-ray CT, and provides a more accurate measure of linear attenuation within a heterogeneous sample.

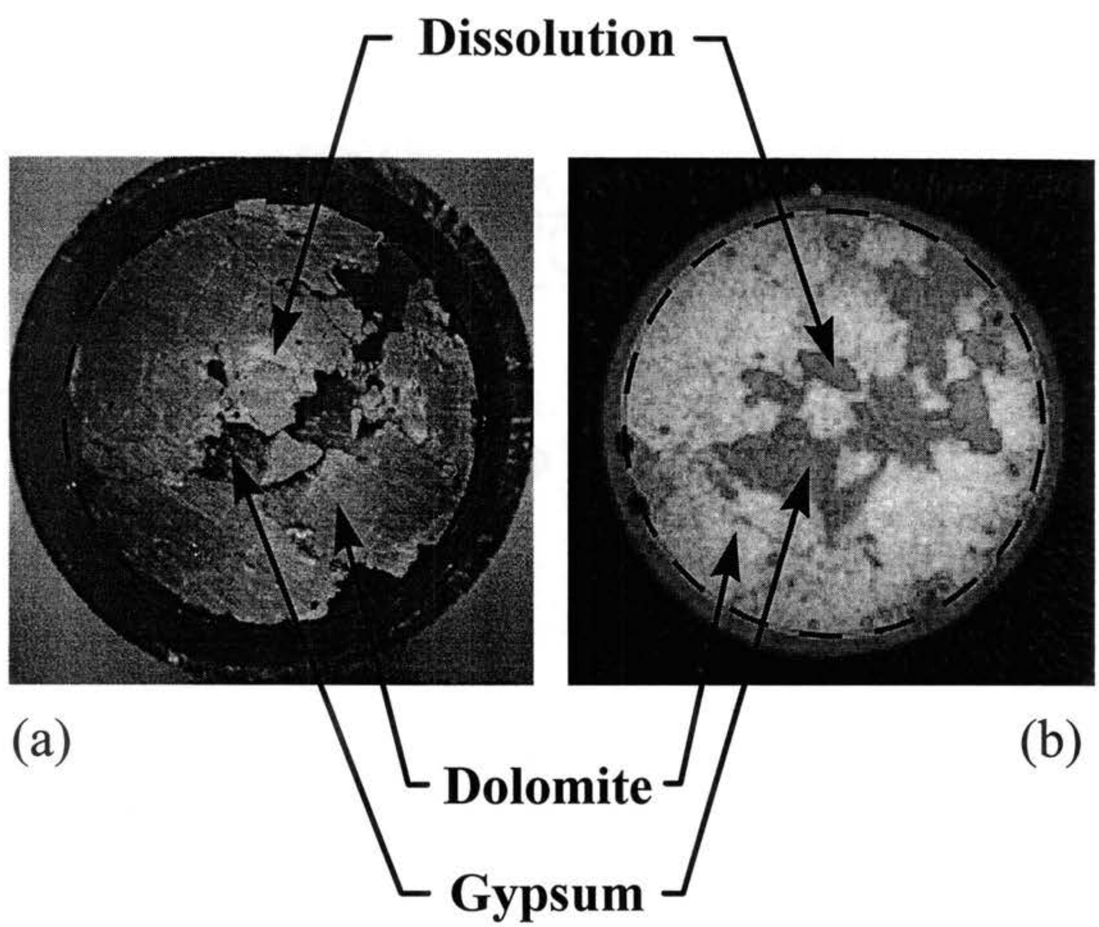


Figure 2-3. Similarity of characteristics of the core sample between (a) the top view of the core, and (b) the reconstructed image of Plane 1, of 3 mm below the top. The attenuation scale on the top left of (b) ranges between -0.0025 and 0.020.

Both the 1.2-Ci ^{137}Cs source and the NaI detector had 50 mm long tungsten collimators with 3 mm diameters. Image scanning density was 120 rays over 120 projections with a ray spacing of 1.5 mm. A total of 31 slices or planes at 3 mm vertical spacing along the axis of the core were imaged. Gamma ray count rates ranged from 30,000 counts/s in air to 2,000 counts/s through the center of the core. With a live detector time of five seconds and a dead time of approximately one second, scanning a single plane took one day. The live time was set to obtain a minimum of 10,000 counts per ray, which provides a maximum standard deviation of 1% of the count. Hardware dead time correction and post-acquisition peak shift detection corrected for count rate dependencies and electronic drift. Using Eq. 2-2, each scan was transformed to a 120 x 120 reconstructed image, producing a voxel size of 1.5 mm on a side and 3 mm high with a volume of 6.8 mm^3. Because of corners and intrusions at the perimeter by rubber, the maximum undisturbed image was 125 mm in diameter, or 5,500 voxels. The region of interest is the volume of the 31 stacked undistributed areas. It has a size of about 1.1×10^6 mm^3, occupying 171,000 voxels, and 69 % of the total core.

This sample conveniently contained its own mineral standard; large, solid gypsum intrusions. Gypsum's density is 2.32 Mg/m^3 [*Weast,* 1988], and its attenuation was measured as 0.0171 mm^{-1}. These values yield a calibration, C = 135.7 mm-Mg/m^3. At 662 keV water has a mass attenuation coefficient 10% greater than most minerals, which will cause a slight over-prediction of density for voxels containing water. The magnitude of any error can be estimated by considering the maximum water content of the sample. Visual inspection showed the secondary porosity was drained, while the primary porosity of the dolomite was water filled. An estimate of the error contributed by water content is the product of water content of 11 %, and the deviation of its attenuation coefficient from the rock minerals of 10 %, which equals 1.1 % or 0.025 Mg/m^3. That error is similar to the uncertainty in the mineral attenuation coefficient.

Results and Discussions

Reconstruction Images

Figure 2-3 provides a comparison of visual features at the top of the core and the reconstructed image of plane 1, 3 mm lower. There is excellent correlation between the discrete regions of dolomite, gypsum, and dissolution features seen in both. Density frequency distributions in the region of interest are plotted for Planes 4, 14, and 30 in Figure 2-4. These planes show the full range in measured density distributions. All planes show a large dolomite peak, but the large gypsum peak in Plane 4 is greatly reduced in Plane 14 and almost disappears in Plane 30. The dolomite density peak occurs at 2.63 Mg/m^3 that corresponds to the wet bulk density previously estimated.

Air-filled porosity distributions are only noticeable in Plane 14 and a few planes near it. The air peak occurs at a slight negative value as a result of the Gibb's effect [*Brown, et al.*, 1993]. This is an indication that most empty pores are just slightly larger than the voxel dimension. The volume of air-filled pores was negligible compared to that of the rest of the materials, and was always less than 1% of the total volume. Considering these two factors it was concluded to neglected the pure air component from the fitting process, but still maintain a zero density minimum for the mixed-component phase.

The variance image from Eq. 2-3 for all planes is similar, a simple topped dome with a maximum value of 0.006 Mg^2/m^6. Figure 2-5 shows the variance image for Plane 1. The fact that the variance is sensitive to the change of density can be noticed by

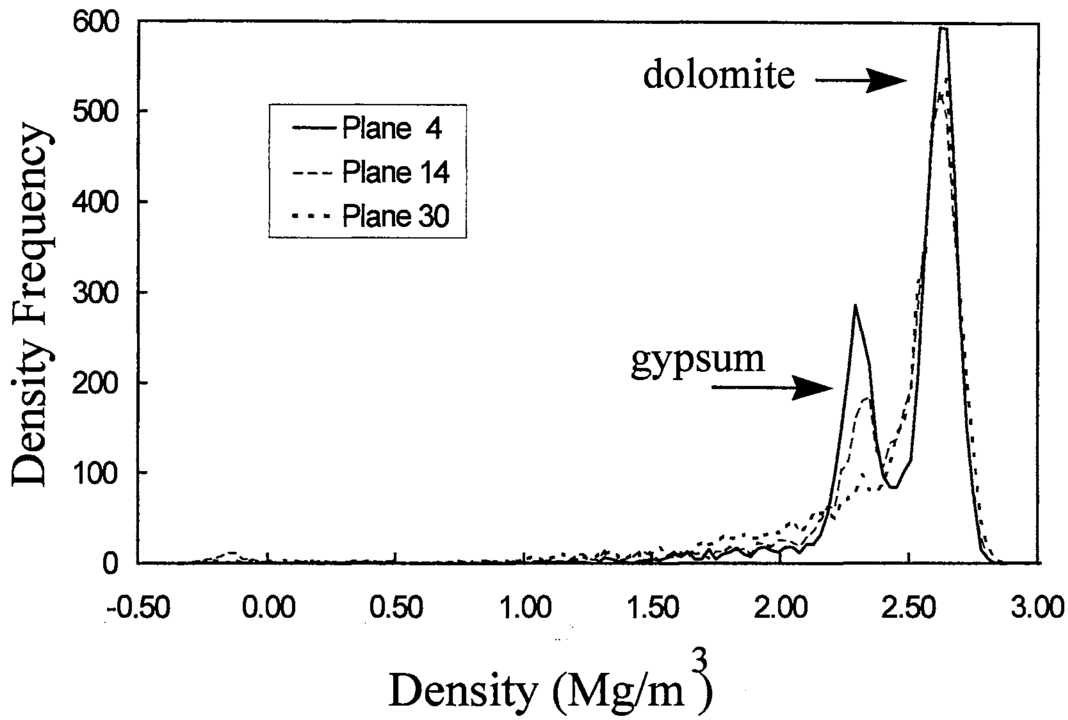


Figure 2-4. Three patterns of density frequency distribution found within the reconstructed image planes.

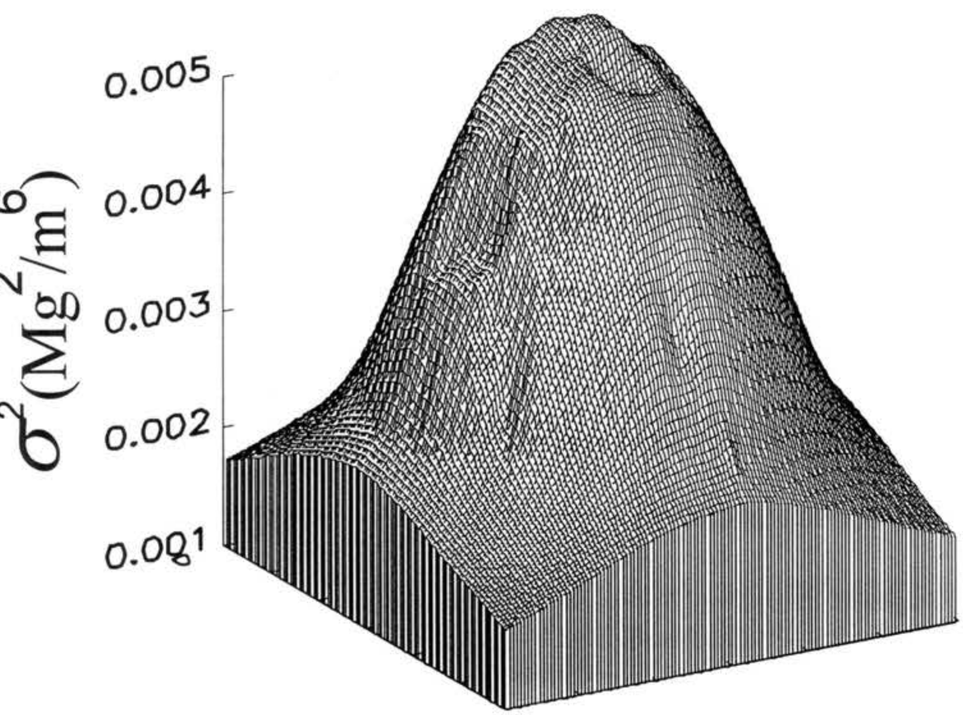


Figure 2-5. Reconstructed variance image for Plane 1.

comparing the two dents on the left hand side of Figure 2-5 with the visible hole feature of Figure 2-3b. The holes in the object increase gamma-ray count and reduce the noise slightly. The photon statistical variance used for the fitting procedure was set to 0.005 Mg^2/m^6 that corresponds to the average value within the region of interest.

Component Distributions

With air-filled porosity neglected, the independent parameters fitted were α, β, R_g, and R_d, where g and d are the subscripts for gypsum and dolomite respectively. Figure 2-6 presents the relative frequency distributions of Plane 4, 14, and 30 and their fitted curves for dolomite, gypsum, and mixed-component phase, while the fitted parameter values for all planes are listed in Table 2-1. The standardized residual error for Planes 4, 14, and 30 is shown in Figure 2-7. Each plot has a near zero sum and no clear trend over the density range. Other planes are similar. The final verification of the fit is provided by applying Eq. 2-17 and 2-18 between 0.50 and 3.38 Mg/m^3. χ_C^2 for each fit is also listed in Table 2-1 and is less than the critical value ($\chi^2_{(0.05,\ 93)}$ = 115 with v = 0.05, J = 98, and i = 4). There is no significant evidence to suggest that the estimated distributions do not provide an adequate fit to the measured data.

Figure 2-8 compares the variation of bulk density and the volume fraction of components along the core length for 31 planes. As can be seen, the distribution among the three components' changes dramatically. While the dolomite content fluctuates in the range of about 40 to 65%, the gypsum content steadily decreases from 25 to near 0%, and the mixed-component phase shows a steady increase from 15 to 50%. Bulk density shown is the average density in each plane, and ranged between 2.20 and 2.43 Mg/m^3.

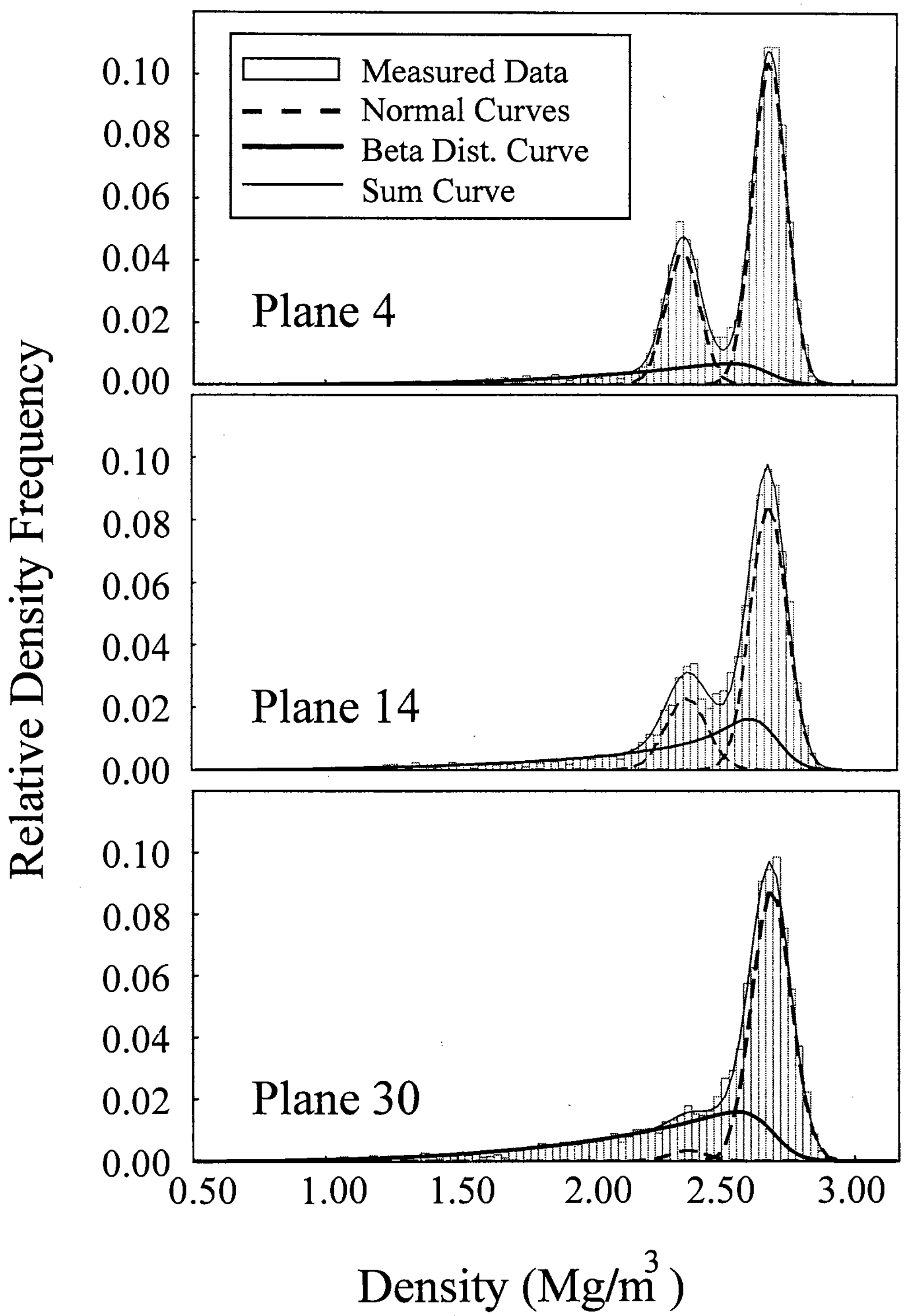


Figure 2-6. Fitted density frequency distributions for Planes 4, 14, and 30.

Table 2-1. Fitting parameters for all 31 planes.

Plane	α	β	R_g	R_d	R_m	χ_c^2*
1	7.00	1.73	0.18	0.66	0.16	107
2	5.60	0.98	0.18	0.58	0.24	109
3	6.90	1.10	0.19	0.56	0.25	78
4	5.80	1.10	0.25	0.59	0.17	101
5	5.40	1.10	0.25	0.59	0.16	88
6	6.30	1.06	0.24	0.58	0.18	103
7	7.00	1.09	0.20	0.60	0.20	80
8	5.70	0.86	0.19	0.59	0.22	77
9	7.80	1.20	0.18	0.63	0.19	105
10	4.40	0.61	0.15	0.61	0.24	96
11	3.50	0.41	0.19	0.56	0.25	86
12	3.25	0.41	0.17	0.53	0.30	81
13	3.30	0.44	0.16	0.52	0.32	75
14	3.69	0.63	0.16	0.52	0.32	106
15	3.35	0.64	0.18	0.51	0.31	97
16	3.80	0.65	0.15	0.49	0.36	89
17	3.90	0.71	0.15	0.51	0.34	98
18	5.25	1.02	0.12	0.51	0.37	86
19	5.99	0.99	0.09	0.52	0.39	88
20	5.40	0.87	0.09	0.50	0.41	111
21	6.00	1.15	0.11	0.52	0.37	111
22	5.30	0.72	0.08	0.39	0.53	112
23	5.10	0.72	0.09	0.39	0.52	113
24	5.40	0.78	0.08	0.48	0.44	104
25	6.30	0.87	0.02	0.54	0.44	106
26	6.60	0.88	0.04	0.54	0.42	101
27	5.00	0.81	0.02	0.48	0.50	112
28	5.40	1.06	0.01	0.52	0.47	104
29	4.80	1.01	0.02	0.54	0.44	106
30	4.70	0.97	0.02	0.58	0.40	96
31	4.15	0.85	0.01	0.60	0.39	113

*Note: $\chi_c^2{}_{0.05} = 116$ for 93 degrees of freedom.

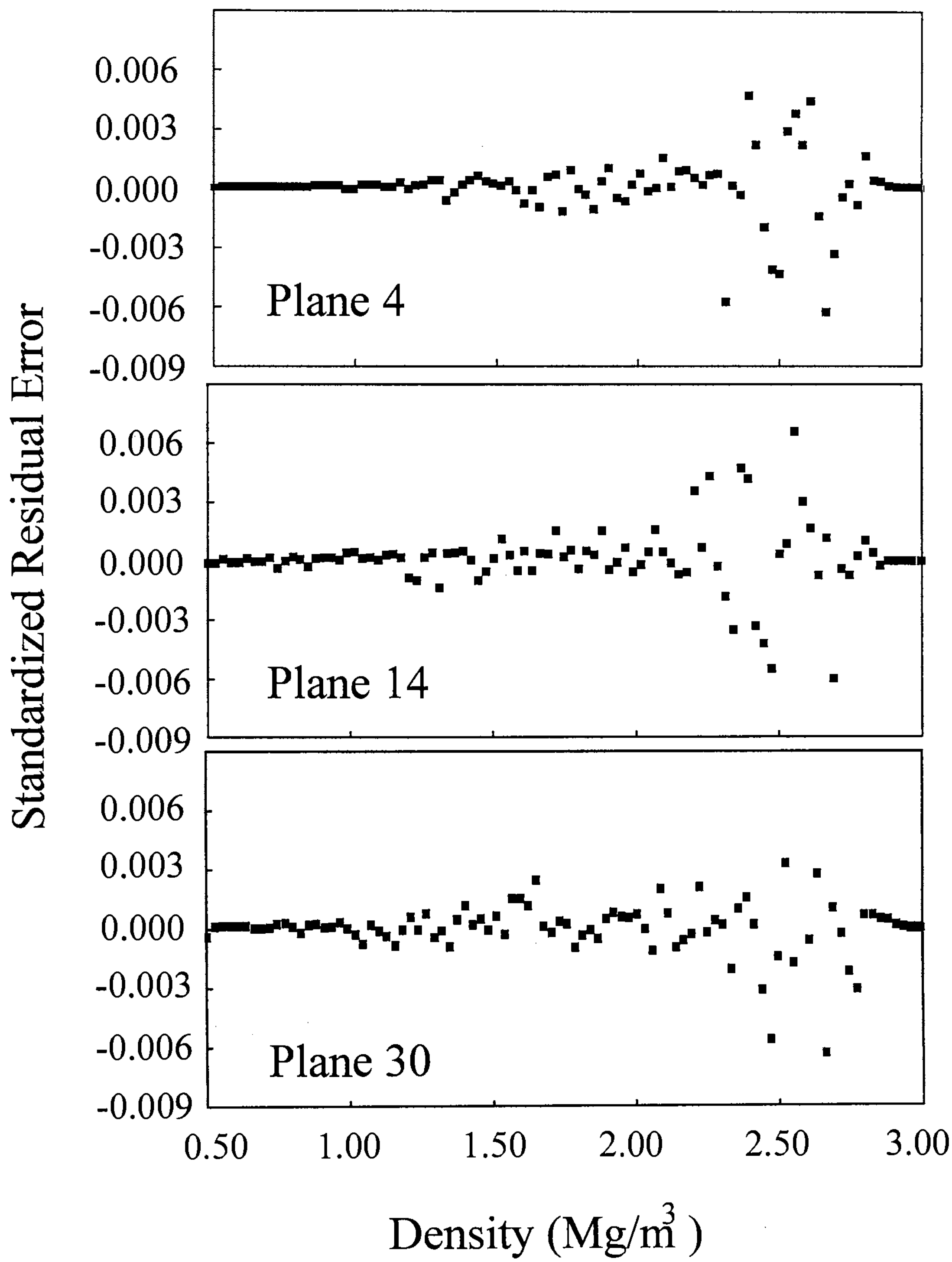


Figure 2-7. Standardized residual plots for fitting of Planes 4, 14, and 30.

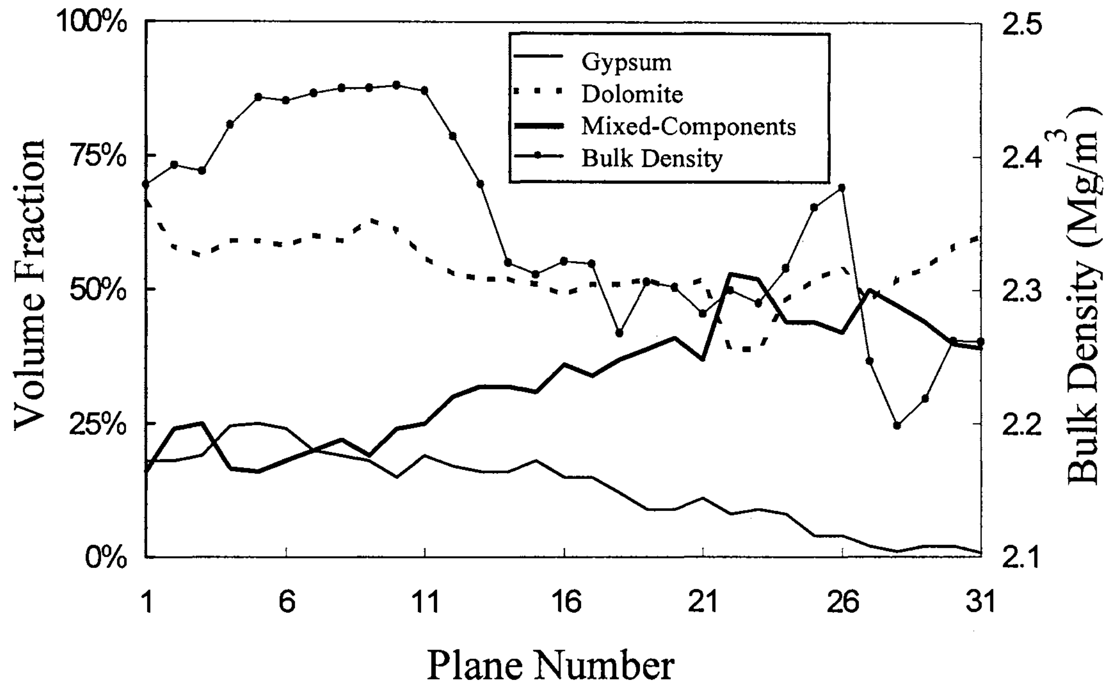


Figure 2-8. Bulk density and volume fraction distribution along the longitudinal axis of the core.

Variations attributed to both the changes of porosity and component fractions are observed in Figure 2-8. Between Planes 6 and 11 the fraction of each individual component changes, but bulk density remains constant. Conversely, bulk density shifts abruptly between 11 and 14, and is characterized by increasing mixed-component phase. Generally, the increase of mixed-component phase reflects increasing macro-porosity. Those increases should indicate a large increase in the hydraulic conductivity of the core along its length. The inverse relationship between gypsum and mixed component phases imply the gypsum is filling a dolomite matrix which initially had a relatively constant macro-porosity. This substantiates *Holt's* [1997] thesis that the Culebra Dolomite porosity at the Waste Isolation Pilot Plant site was completely open at some point in the past.

The accuracy and uncertainty of the volume estimates made here can be determined using stochastic procedures such as Bayesian statistical inferences [*Yue*, 1994]. However, that is well beyond the scope of this work. Likewise, the suitability of the demonstrated method to separated components with smaller attenuation differences, or to be used with different CT systems will require its application to specific cases.

Conclusions

Measured CT density frequency distributions are shown to be a convolution of the true component contents and the Gaussian measurement error resulting from photon statistics. By using the known photon statistical errors and simple parameter estimation it is possible to decompose the measured continuous density frequency distributions to true component contents. Each resulting estimated density frequency distribution is a deterministic compositional measurement of the sample that can be justified with statistical measures.

Any fitted density distribution of a porous medium sample is made up of the principal mineral and solid-rock components and a residual of mixed-component phase. The magnitude of the density frequency distribution for the mixed-component phase indicates the amount of component mixing at the voxel scale, and provides a means to quantify heterogeneity. In this sample the dolomite and primary porosity appeared as a single phase, indicating the primary porosity is well below the voxel size. Conversely, the secondary porosity produced a small peak density frequency distribution and a wide distribution of mixed-component phase indicating its scale ranges down from a few millimeters.

The ability to separate peaks in the density frequency distribution is controlled by the peak densities, component volume fractions and the degree of voxel-scale sample heterogeneity. Frequency distributions with large density differences and equal component volumes allow the best frequency peak separation. Increasing component heterogeneity or the volume fraction of the mixed-component phase blurs peak shapes. With the help of this deconvolution algorithm, the component distributions of CT images were adequately defined even though the density difference between peaks was only 15% and one component decreased to near zero volume. The potential of this method for quantifying the content of multiple solid and fluid phases with smaller density and volume contrast is worth exploring on this and other CT systems.

References

Anderson, S. H., and J. W. Hopmans (Ed.), *Tomography of Soil-Water-Root Processes,* Soil Sci. Soc. Am. Special Publication No. 36, Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Brown, G. O., M. L. Stone and J. M. Gazin, Accuracy of gamma ray computerized tomography in porous media, *Water Resour. Res.*, 29(2), 479-486, 1993.

Crestana, S., S. Mascarenhas, and R. S. Pozzi-Mucelli, Static and dynamic three-dimensional studies of water in soil using computed tomographic scanning, *Soil Sci.*, 140(5), 326-332, 1985.

Gardner, W., Water content, in *Methods of Soil Analysis, Part 1 Physical and Mineralogical Methods* - Agronomy Monograph no. 9, 2nd ed., Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, pp. 493-544, 1986.

Haan, C. T., *Statistical Methods in Hydrology*, pp. 174-178, The Iowa State University Press, Ames, 1977.

Hainsworth, J. M., and L. A. G. Aylmore, The use of computer-assisted tomography to determine spatial distribution of soil water content, *Aust. J. Soil Res.*, 21(4), 435-443, 1983.

Holt, R. M., *Conceptual Model for Transport Processes in the Culebra Dolomite Member, Rustler Formation*, SAND97-0194, Sandia National Laboratories, Albuquerque, New Mexico, 1997.

Hopmans, J. W., M. Cislerova and T. Vogel, X-ray tomography of soil properties, in *Tomography of Soil-Water-Root Processes*, edited by S. H. Anderson and J. W.

Hopmans, pp. 17-28, SSSA Special Publication No. 36. Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Kak, A. C., and M. Slaney, *Principles of Computerized Tomographic Imaging,* pp. 8-9, pp. 194-197, IEEE Press, New York, 1988.

Kantzas, A., Investigation of physical properties of porous rocks and fluid flow phenomena in porous media using computer assisted tomography, *In Situ*, 14(1), 77-132, 1990.

Kantzas, A., D. F. Marentette and K. M. Jha, Computer-assisted tomography: from quantitative visualization to quantitative core analysis, *J. Can. Petrol. Tech.*, 31(9), 48-56, 1992.

Kelley, V. A., and G. J. Saulnier, Jr., *Core Analysis for Selected Samples From the Culebra Dolomite at the Waste Isolation Pilot Plant Site*, SAND90-7011, Sandia National Laboratories, Albuquerque, New Mexico, 1990.

Lucero, D. A., F. Gelbard, Y. K. Behl, and J. A. Romero, Test Plan for Laboratory Column Experiments for Radionuclide Adsorption studies of the Culebra Dolomite Member of the Rustler Formation at the WIPP site, WIPP test plan, *TP95-03*, Sandia National Laboratories, Albuquerque, New Mexico, 1994.

Luo, X., and L. G. Wells, Evaluation of gamma ray attenuation for measuring soil bulk density: Part 1. Laboratory Investigation, *Trans. of Am. Soc. Ag. Eng.*, 35(1), 17-26, 1992.

Mendenhall, W., D. D. Wackerly, and R. L. Scheaffer, *Mathematical Statistics with Applications*, 4th ed., pp. 171-175, Duxbury Press, Belmont, California, 1990.

Orsi, T. H., C. M. Edwards and A. L. Anderson, X-ray computed tomography: a nondestructive method for quantitative analysis of sediment cores, *J. Sedi. Res.*, A64(3), 690-693, 1994.

Port, S. C., *Theoretical Probability for Applications*, Chapter 36, John Wiley & Sons, Inc., New York, 1994.

Petrovic, A. M., J. E. Siebert, and P. E. Rieke, Soil bulk density analysis in three dimensions by computed tomographic scanning, *Soil Sci. Soc. Am. J.*, 46(3), 445-450, 1982.

Peyton, R. L., C. J. Gantzer, S. H. Anderson, B. A. Haeffner and P. Pfeifer, Fractal dimension to describe soil macropore structure using x-ray computed tomography, *Water Resour. Res.*, 30(3), 691-700, 1994.

Shepp, L. A., and B. F. Logan, The fourier reconstruction of a head section, *IEEE Trans. Nucl. Sci.*, 21(3), 21-43, 1974.

Spanne, P., K. W. Jones, L. Prunty, and S. H. Anderson, Potential applications of synchrotron computed microtomography to soil science, in *Tomography of Soil-Water-Root Processes*, edited by S. H. Anderson and J. W. Hopmans, pp. 43-58, SSSA Special Publication No. 36. Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Weast, R. C. (Ed.), *Handbook of Chemistry and Physics*, pp. F-1, 1st Student Edition, CRC Press, Inc., Boca Raton, FL, 1988.

Yue, J., *Parameter Estimation and Uncertainty for Volatile Organic Transport*, Ph.D. Thesis, Biosystems Engineering, Oklahoma State University, Stillwater, 185 pages, 1994.

Chapter III

QUANTIFICATION OF POROUS MEDIA USING GAMMA RAY TOMOGRAPHY AND STATISTICAL SEGREGATION THRESHOLD

Abstract

A computerized tomography (CT) statistical segregation threshold (SST) that determines spatial distribution of solid and pore components in a laboratory-scale porous media is presented. The proposed algorithm combines basic image processing techniques with consideration of measurement errors and CT scanning resolution. While simple threshold methods concentrate on identifying pore features in CT images, the SST characterizes both volume fraction and spatial distribution of multiple pure components. This provides a major improvement when pores are smaller than the image resolution. By dividing the original CT image into mixed-component, peak- and tail-voxel volume fractions, the proposed algorithm provides a semi-automatic and theoretical sound technique for component segregation using CT images. The use of a local threshold value in SST algorithm shows the segregation results are insensitive to the threshold value. Four acrylic test objects are used to illustrate and test the ability of the SST.

Introduction

The composition and structure of porous media dominate groundwater flow and contaminant transport phenomena. Traditional destructive methods measure interior structure of porous media at the expense of loss of sample, and are often impractical for full three dimensional image analysis and later verification. Computerized tomography (CT) has been used to overcome these problems and provides the same or better understanding of interior physical properties [*Anderson and Hopmans,* 1994].

CT images have been used extensively to quantify features, such as macropores and fractures [*Anderson et al.*, 1990; *Hopmans et al.,* 1994; *Warner et al.*, 1989; *Warner et al.*, 1991; *Greves et al.*, 1989]. *Peyton et al.* [1991] developed an iterative procedure that considers macropore spatial relationship and evaluates variously sized macropores. *Kantzas* [1991] compared the bulk density and effective atomic number to discriminate sulfur within a dolomitic rock. However, most quantified estimates of component contents and feature properties in CT imaging are determined by simple threshold methods [*Spanne et al.*, 1994; *Kantzas*, 1990; *Kantzas et al.*, 1991]. Reconstructed voxel attenuation values reflect photon statistical errors, sample heterogeneity and CT scanning resolution [*Brown et al.*, 1993]. The statistical frequency distributions of pure components may overlap depending on component attenuation and volume fraction. The previous chapter demonstrated a histogram deconvolution algorithm to quantify component contents and spatial heterogeneity of CT images. To quantify entire volume fraction of pure components, the mixed-component phase has to be further defined.

The objectives of the study are to quantify the character of mixed-component voxels in sample heterogeneity and to develop a comprehensive component segregation algorithm for CT images suitable for porous media applications.

Statistical Segregation Threshold

Voxel Statistical Classification

The CT image of a single component object will follow a simple Gaussian distribution. However, scanning a heterogeneous sample under a limited scanning resolution will produce a complex voxel frequency distribution. In this study, voxels that include multiple components are identified as mixed-component voxels. The attenuation of a mixed-component voxel, $\mu_m^*(x,y)$, at position (x,y) can be defined as a summed result that satisfies

$$\mu_m^*(x,y) = \sum_{k=1}^{K} r_k(x,y)\mu_k^* \quad (3\text{-}1)$$

where μ_k^*, and $r_k(x,y)$ are the attenuation and volume fraction of the pure component k from a total of K-component image array. It follows that $\sum_{k=1}^{K} r_k = 1$. The true attenuation distribution function, $f(\mu)$, for a K component system is given by

$$f(\mu) = \sum_{k=1}^{K} R_k(\mu)Q_k(\mu) + R_m(\mu)f_m^*(\mu') \quad (3\text{-}2)$$

where R_k and R_m are the volume fractions of the pure component k and the mixed-component phase, $Q_k(\mu)$ is an impulse function, and $f_m^*(\mu')$ is the Beta distribution which characterizes the frequency distribution of the mixed-component voxels. The normalized attenuation, μ', is equal to $[\mu / \mu_{\max}]$ where μ_{max} is the maximum attenuation in the frequency distribution. Continuity requires $R_k, R_m \geq 0$ and $\sum_{k=1}^{K} R_k + R_m = 1$. The measured distribution function, $h(\mu)$, is given by the convolution of the true density distribution, $f(\mu)$, with the Gaussian error distribution, $g(\mu)$;

$$h(\mu) = f(\mu) * g(\mu) \tag{3-3}$$

where * indicates the convolution of $f(\mu)$ with $g(\mu)$. Therefore, for a two component system, the measured frequency distribution can be decomposed into two simple Gaussian distributions and one mixèd-component curve as shown in Figure 3-1. As convoluted by the Gaussian error distribution pure components assume a Gaussian bell shaped distribution. In this study, voxels distributed one standard deviation to both sides of the pure component mean are categorized as peak voxels while the rest of the distribution are categorized as tail voxels. With the above definition, a 2-D CT image array, U, may be divided into three volume fractions, mixed-component, U_M, peak-voxels, U_L, and tail voxels, U_H. This may be expressed as

$$U = U_M + U_H + U_L \tag{3-4}$$

Segregation Procedures

<u>Voxel Type Identification</u>

The distinctness of the three voxel types indicated in Eq. 3-4 intuitively leads to the development of the SST, based on both the statistical and the spatial distribution of voxels. The algorithm is performed in three steps, first Sobel edge detection is applied to locate mixed-component voxels on the boundaries between pure components. Second, peak voxels are identified based on the mean attenuation and image standard deviation.

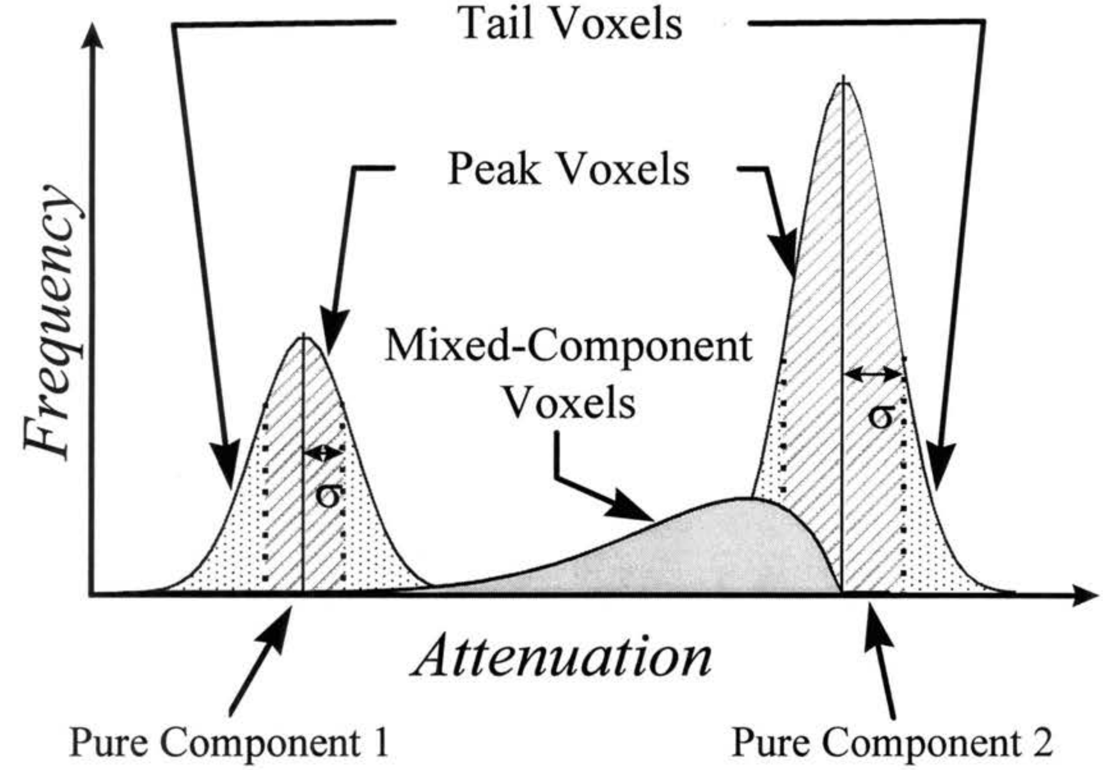


Figure 3- 1. Conceptualized voxel classification.

Any remaining voxels are classified as tail voxels. Finally, using a nearest neighborhood technique, the mixed-component and tail voxels are assigned a component identity consistent with the majority of its nearest neighbors.

If the attenuation difference of pure components in the raw CT image is low, the edge detection operation becomes less sensitive. An exponential transformation of raw attenuation can help to enforce the detection capability. The new image array, F, after applying the exponential transform, is written as

$$F = exp[U] \tag{3-5}$$

F is used either exponentially transformed or equivalent to U, as a representative array domain of interest for the following discussion. Eq. 3-4 is redefined as,

$$F = F_M + F_H + F_L \tag{3-6}$$

According to the characteristic of mixed-component voxels defined in Eq. 3-1, they are expected to be observed along the component boundaries, or edges in traditional image processing terms. The computation of the attenuation gradient of an image, or derivative, may be implemented in digital forms in several ways. Roberts edge detection that is less sensitive to the highly heterogeneous rock mass, was found inferior to Sobel detection. Therefore, Sobel edge detection that provides both differencing and smoothing effects, was selected as our edge operation [*Gonzales and Woods*, 1993].

The edge voxel gradient, $G_x(x,y)$ and $G_y(x,y)$ in x and y axis directions at (x, y), are given by

$$G_x(x,y) = (F_{x-1,y+1} + 2F_{x,y+1} + F_{x+1,y+1}) - (F_{x-1,y-1} + 2F_{x,y-1} + F_{x+1,y-1}) \tag{3-7}$$

and

$$G_y(x,y) = (F_{x+1,y-1} + 2F_{x+1,y} + F_{x+1,y+1}) - (F_{x-1,y-1} + 2F_{x-1,y} + F_{x-1,y+1}) \tag{3-8}$$

The total edge voxel gradient, *G(x,y)*, may be approximated by the summation of Eqs. 3-7 and 3-8,

$$G(x,y) \approx |G_x(x,y)| + |G_y(x,y)| \tag{3-9}$$

The approximation of Eq. 3-9 allows for fast integer calculation. More details can be found in *Gonzales and Woods* [1993]. The mixed-component array, F_M, in Eq. 3-6 is therefore determined by

$$F_M(x,y) = \begin{cases} F(x,y) & \textit{if } G(x,y) > t \\ 0 & \textit{elsewhere} \end{cases} \tag{3-10}$$

where t is the edge threshold. The determination of edge threshold is an important issue during the segregation procedure and will be discussed in the later section. A new image array *F'*, that excludes those mixed-component fraction array, F_M, is generated.

As previously defined, peak-voxel attenuation falls within one standard deviation of the pure component mean given by

$$F_L(x,y) = \begin{cases} F'(x,y) & \overline{\mu_k} - \sigma \le F'(x,y) < \overline{\mu_k} + \sigma \quad k = 1 \textit{ to } K \\ 0 & \textit{elsewhere} \end{cases} \tag{3-11}$$

The tail-voxel fraction array, F_H, is therefore,

$$F_H = F - F_M - F_L \tag{3-12}$$

Component Segregation

A component voxel that is defined from the peak-voxel fraction array, is expressed as

$$CS_L(F_L(x,y)) = \begin{cases} k & F_L(x,y) \in PC_k \quad k = 1 \text{ to } K \\ 0 & \text{elsewhere} \end{cases} \tag{3-13}$$

where CS_L represents the component segregation function for the peak-voxel array that assigns component identity, k, to measured voxels if their attenuation falls within the k^{th} Gaussian distribution function, PC_k. The component identity from those mixed-component fraction array at position (x,y), is given by

$$CS_M(F_M(x,y)) = \begin{cases} k & k \in SC \\ 0 & \text{elsewhere} \end{cases} \tag{3-14}$$

CS_M is the component segregation function for mixed-component voxels. The spatial correlation function, SC, is used to determine voxel identity according to the correlation between the investigated voxel and its surrounding components and can be further elaborated as

$$SC(F_M(x,y)) = k \qquad \text{if } VF_k > VF_l \tag{3-15}$$

where $l = 1 \text{ to } K$ with $l \neq k$, and the k^{th} component with the largest volume fraction, VF, at $F_M(x,y)$'s surrounding neighborhood is then assigned as the component of $F_M(x, y)$.

The same procedure used for segregating the component content of the mixed-component class is also applied to segregate the tail-voxel array and is expressed as

$$CS_H(F_H(x,y)) = \begin{cases} k & k \in SC \\ 0 & \text{elsewhere} \end{cases} \tag{3-16}$$

Combining the results from Eqs. 3-13, 3-14 and 3-16, the post-processed component segregated array, CS, is given by

$$CS = CS_H + CS_L + CS_M \tag{3-17}$$

The volume fraction of each component is defined as the ratio of the component voxels to the total voxels.

Materials and Methods

CT System

A parallel beam gamma ray CT scanner with a 1.2-Ci ^{137}Cs source is used in this study [*Brown et al.*, 1993]. Replaceable source and detector collimators allow varied scanning resolution and three collimation sizes, 1.5-, 2- and 3-mm, were selected for this study. The live times used for the three collimations are 40, 10 and 3 seconds and produced approximately the same unattenuated count of 90,000 per ray. The ray step sizes for 1.5-, 2- and 3-mm collimations are 0.75-, 1- and 1.5-mm, respectively. To obtain an equal scanning area of 3600 mm^2, the time required for scanning a single plane using 3-mm collimation was 3 hours, for the 2-mm collimation it was 12 hours and for the 1.5-mm collimation it was approximately 3 days. Measured attenuation coefficients were calibrated with the use of water filled standards to produce water attenuation of 0.00859 mm^{-1} at 23 degree C.

Test Object Cylinders

Four 50-mm diameter acrylic cylinders with known hole features and spatial distributions are used as shown in Figure 3-2. In Figure 3-2 Object 1 has one 25-mm diameter hole on the center, while Object 2 has 25 5-mm diameter holes evenly distributed within the object. Void space in Objects 1 and 2 is 25% of the entire cylinder region. Figures 3-2c and 3-2d display the 201, 2-mm and 481, 1-mm holes in Objects 3 and 4, respectively. Void volume fractions in Objects 3 and 4 are 32 and 19%.

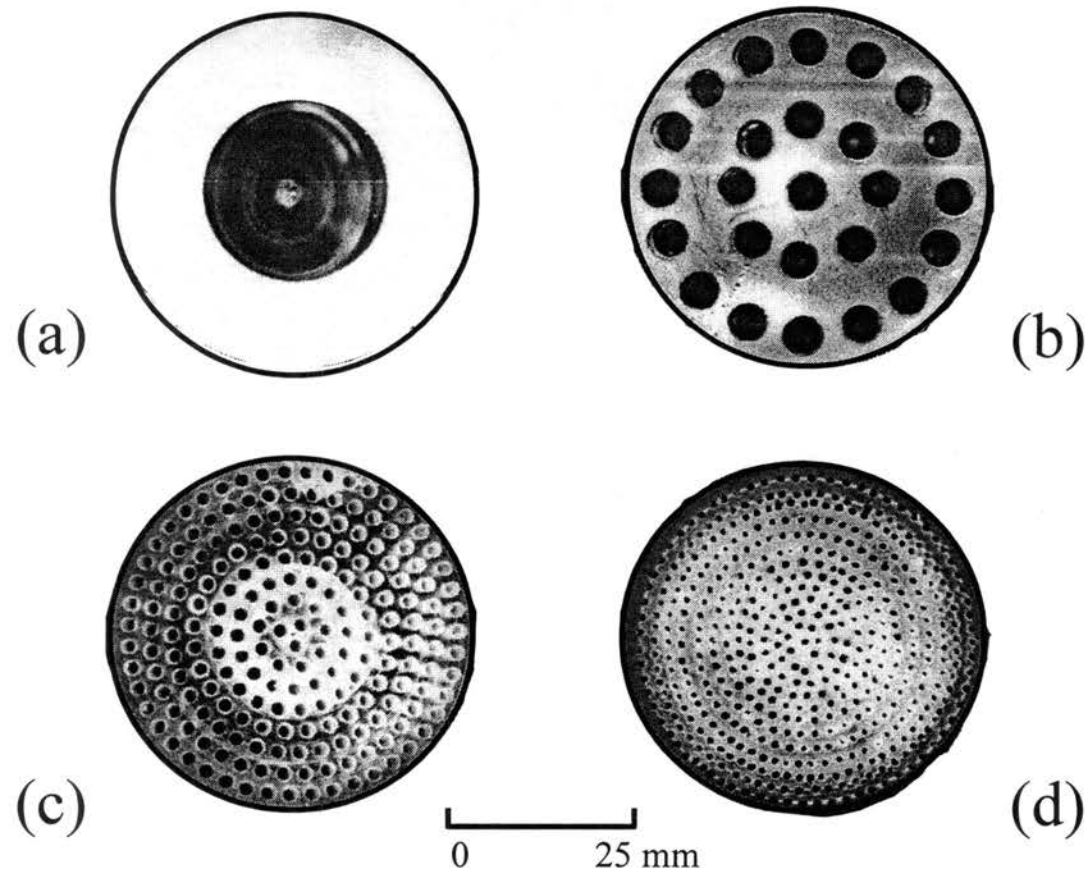


Figure 3- 2. Four 50-mm diameter acrylic cylinders with hole features; (a) Test Object 1; (b) Test Object 2; (c) Test Object 3; (d) Test Object 4.

Objects 1 and 2 were designed for estimating properties of mixed-component voxels, while Objects 3 and 4 are designed to quantify the correlation between scanning resolution and procedure performance. The acrylic has an attenuation coefficient of 0.0098 mm^{-1}.

By introducing water into some of the void spaces of the test objects, a three-phase system is created. In this study, de-ionized water at a room temperature of 23 degree C is used as the third pure component phase, which has a uniform attenuation coefficient of 0.0086 mm^{-1}.

Simple Threshold Method

The most frequently used method in CT image segregation is the simple threshold algorithm that will be used for comparison. For a K component system, the simple threshold component function, *STC*, is given by

$$STC(F(x,y)) = \begin{cases} k & \text{if } T_{k-1} \le F(x,y) \\ 1 & \text{if } F(x,y) < T_1 \\ m & \text{if } T_{m-1} \le F(x,y) < T_m \quad \text{where } m = 2 \text{ to } k-1 \end{cases} \tag{3-18}$$

Under the consideration of the two-component system, water and acrylic, the acrylic-water threshold, T_1, is set to the mean attenuation of the two components, or 0.0092 mm^{-1}, while the water-void threshold, T_2, is defined at the location three standard deviations from the mean attenuation of water, or about 0.0070 mm^{-1}.

Results and Discussion

Test Object Images

Both two- and three-component test object images are used in the study. Figure 3-3 presents the corresponding CT images from Objects 1 to 4 in Figure 3-2. In Figures 3-3a and 3b holes in Objects 1 and 2 are well preserved in both solid-void boundaries and the hole shape. However, in Figures 3-3c while Object 3 shows distinguishable holes, the whole shape is ill-defined. When further decreasing hole diameter to 1-mm diameter, both hole feature and shape can not be recognized as shown in Figures 3-3d. It seems that visual inspection of CT images provides a straightforward assessment of possible component contents. However, the identification process requires experience, judgment and time.

SST Results

Figure 3-4 shows a frequency distribution using Object 2 under 1.5-mm collimators and parameter selection for three voxel type fractions, pure component means and their standard deviation. Peak-voxel attenuation coefficients for solid acrylic and void space are zero and 0.0098 mm^{-1}, respectively. An average variance of 2.6×10^{-7} mm^{-2} are used. Figures 3-5a to 3-5c show from left to right the original CT image, binary voxel type images and the SST image.

Three resulting component images using Object 2 under 1.5-, 2- and 3-mm collimation sizes are shown in Figures 3-6. Well segregated feature size and shape are shown in Figures 3-6a and 3-6b, while the resulting image in Figure 3-6c under 3-mm

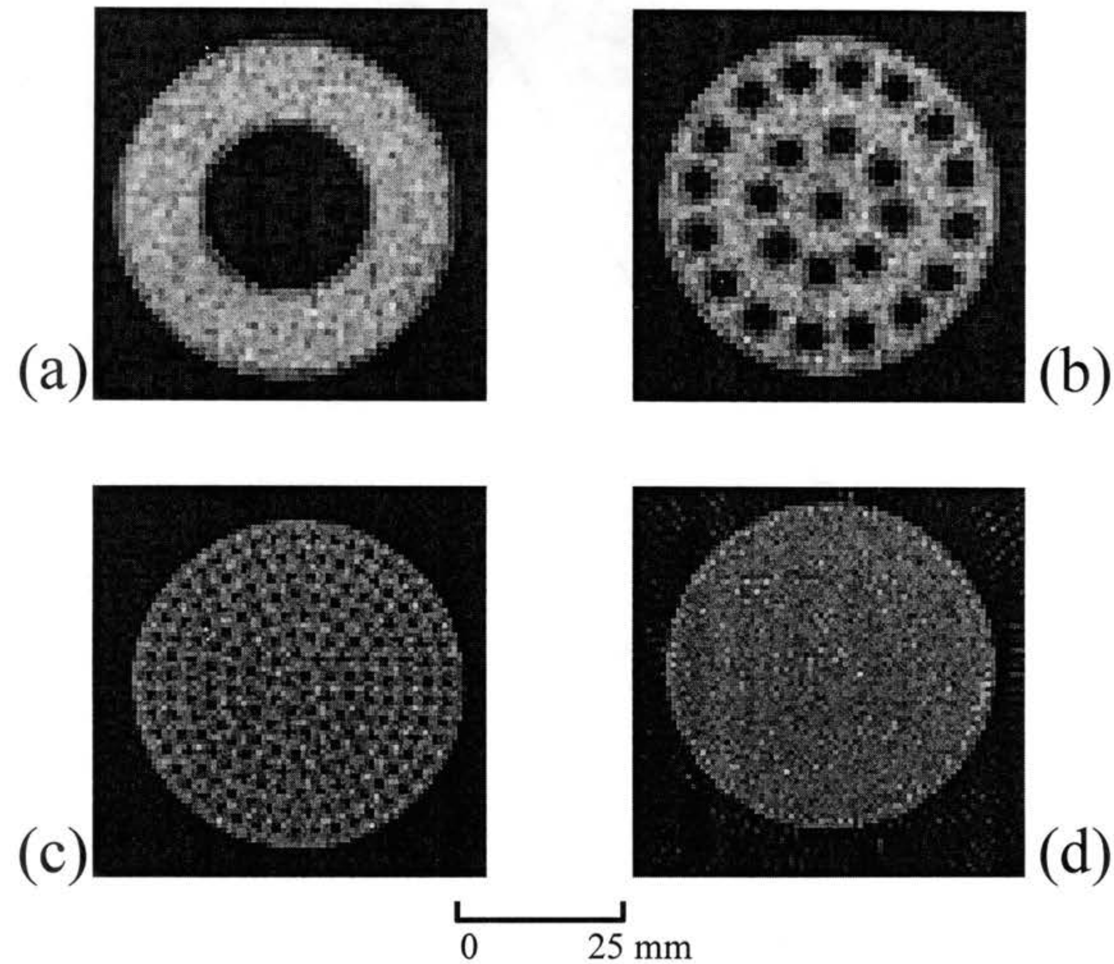


Figure 3- 3. CT images; (a) Test Object 1; (b) Test Object 2; (c) Test Object 3; (d) Test Object 4. Images (a) and (b) use 2-mm collimation at 60 by 60 reconstruction resolution and images (c) and (d) use 1.5-mm collimation at 80 by 80 resolution.

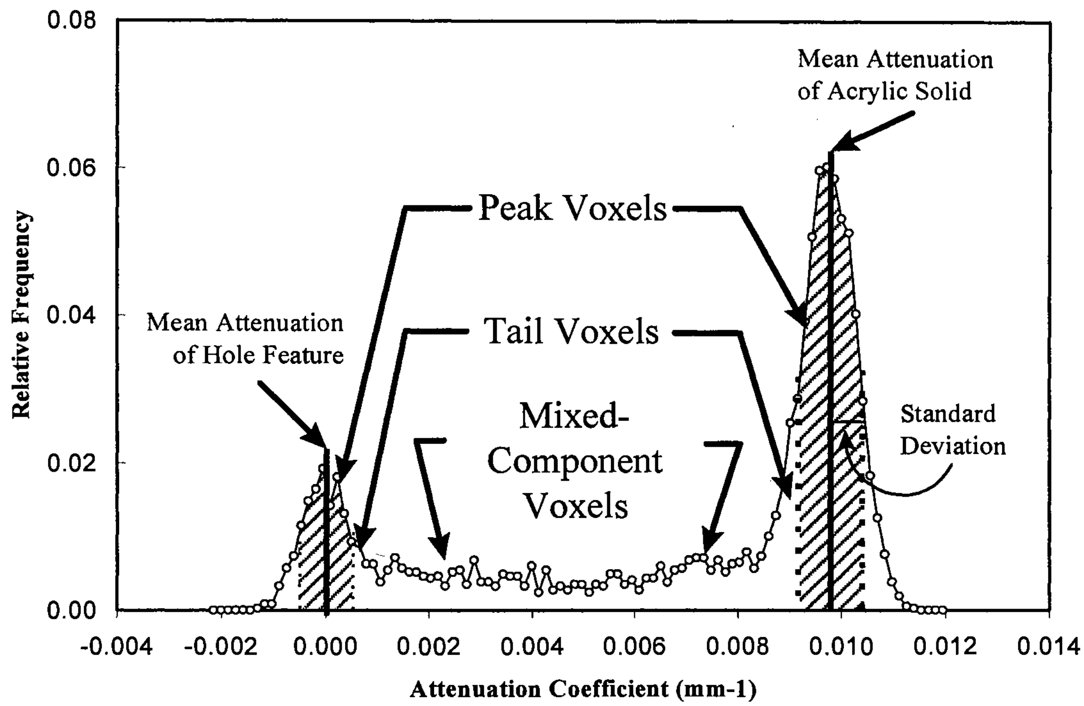


Figure 3-4. Attenuation frequency distribution curve of Object 2 under 1.5-mm collimation. Corresponding segregation parameters, voxel types, peak attenuation coefficient location and average variance range, are identified.

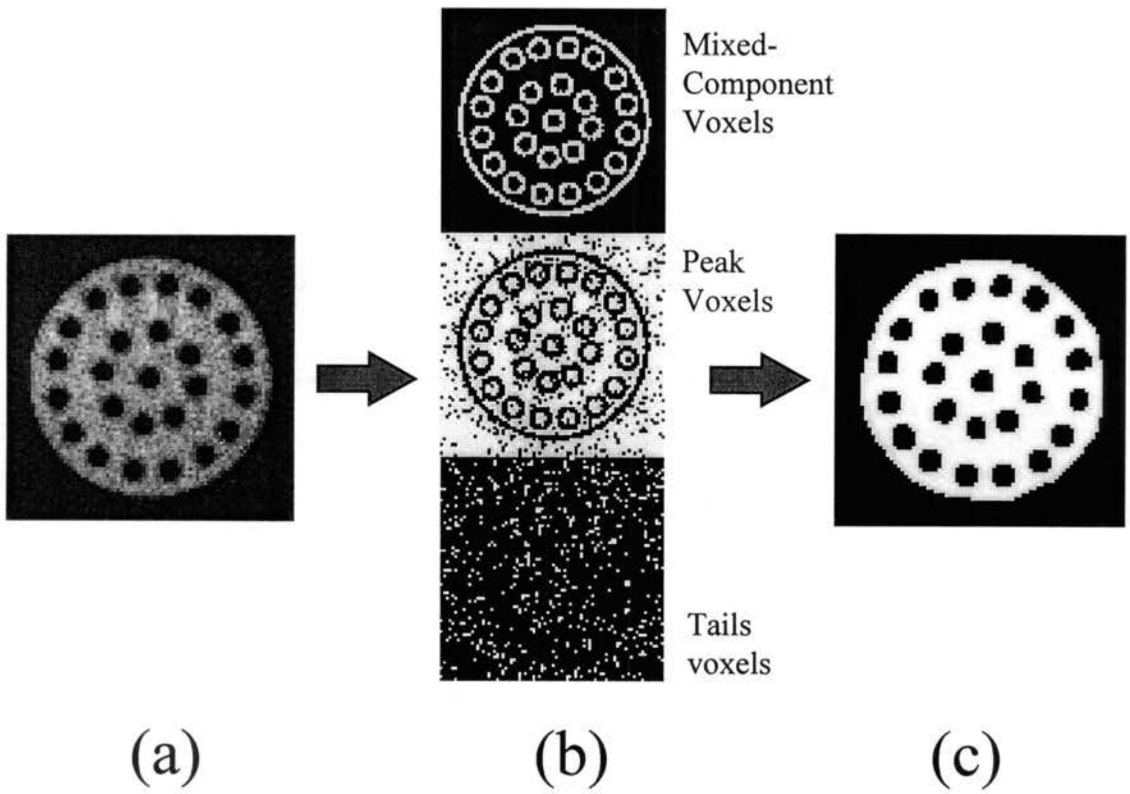


Figure 3-5. Steps in the SST algorithm; (a) Original CT image; (b) Three defined voxel group images; (c) Component segregated image.

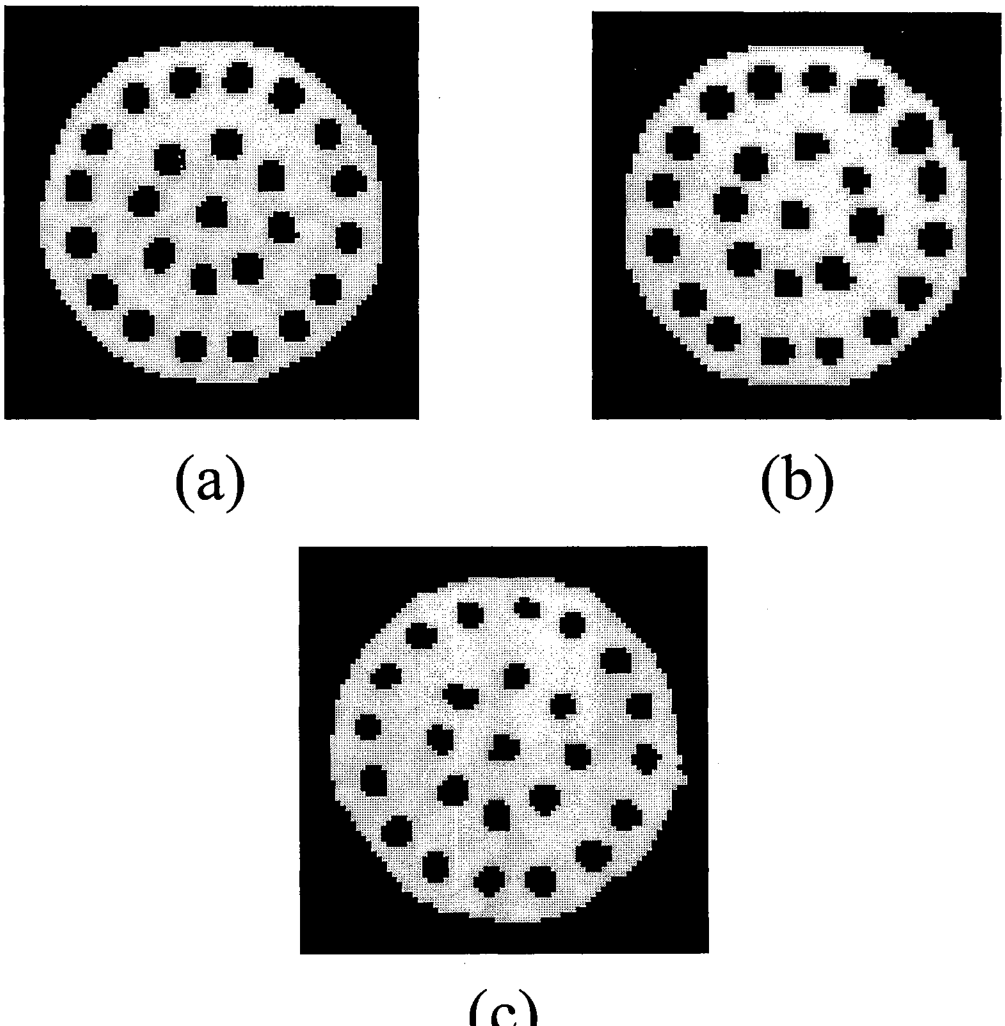


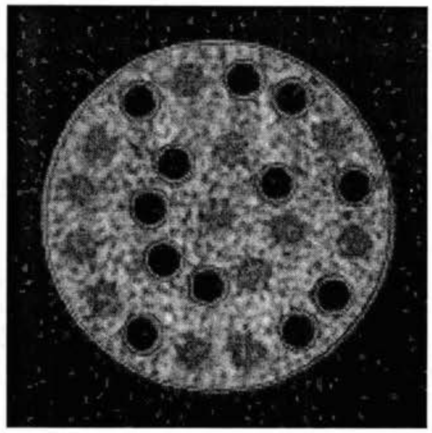
Figure 3-6. Component image using various collimations; (a) 1.5-mm; (b) 2-mm; (c) 3-mm.

resolution shows slight shape defective. Low scanning resolution and the obscure peak pattern reduce the sensitivity for searching peak voxels.

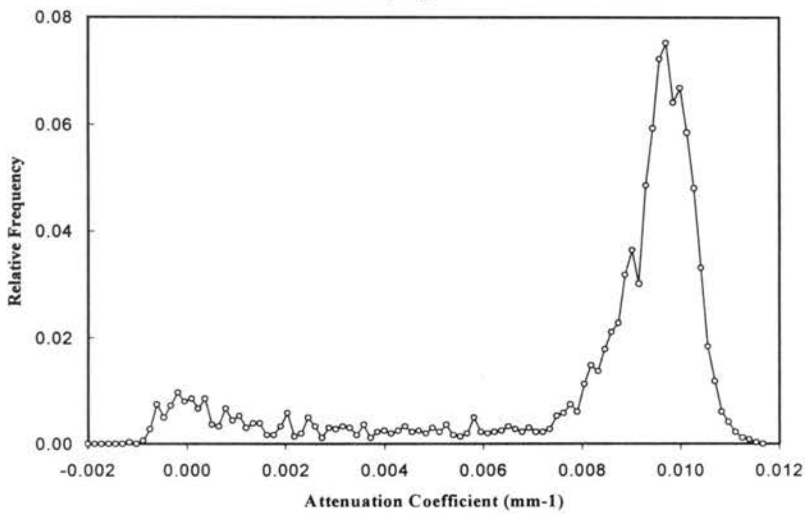
Figure 3-7 shows an original CT image with 13 % water, 12 % void and 75 % acrylic under 1.5-mm collimation scanning and the resulting frequency distribution. The distribution shows only two peaks with the higher peaks containing both water and acrylic voxels. Under such circumstance, the pure component mean from the water is still visible and may be determined by calculating the average attenuation from a large area. Figure 3-8 compares the component images by simple threshold method and the proposed algorithm. The large void component is well defined in both methods, while an overestimated water component in Figure 3-8a is defined by the simple threshold method. Figure 3-8b shows the SST conservatively characterized the water filled voids with less scattering. Due to the specific design to extract mixed-component voxels from the main image body, the mid-attenuation ring shapes caused by the miss treatment of those voxels, is eliminated. The cautiously defined water component voxels, though is smaller than the actual volume size, accurately preserve their spatial distribution.

Sensitivity Analysis and Limitations of the Algorithm

Figure 3-9 displays images generated by the simple threshold method and the SST algorithm under 2-mm collimation scanning using Object 2. The deterioration of the image using the simple threshold is clearly seen while voids are clear, water voxels become more miss-connected. Meanwhile, using SST the identified water bodies maintain their proper spatial distribution, but are somewhat smaller.

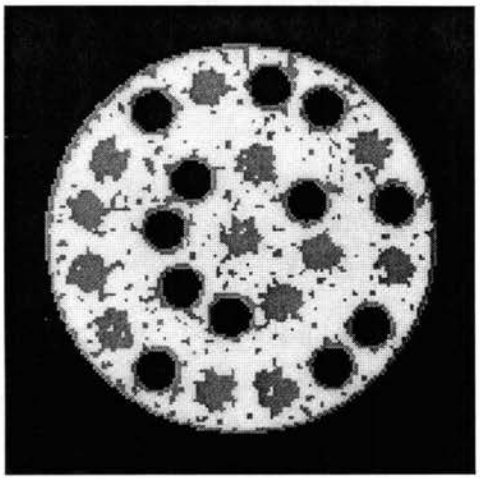


(a)

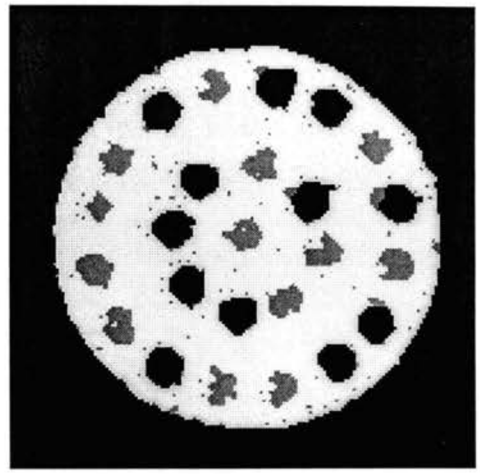


(b)

Figure 3-7. Test Object 2 with one-half of the porosity water filled using 1.5-mm collimation; (a) Original CT image; (b) Frequency distribution of the original image.

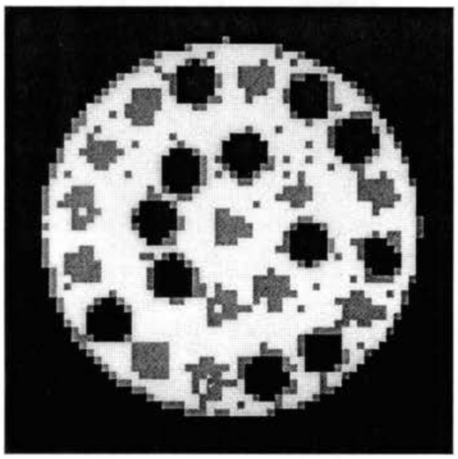


(a)

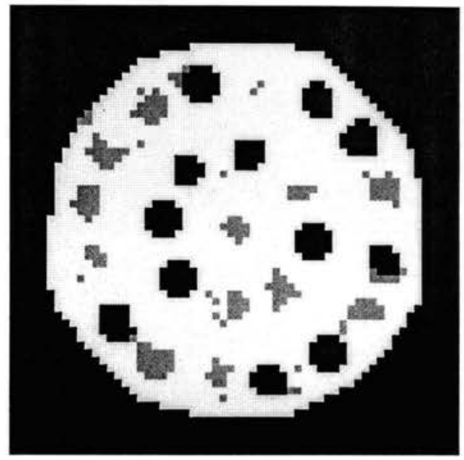


(b)

Figure 3-8. Component images; (a) the simple threshold; (b) the SST.



(a)



(b)

Figure 3-9. Component results for Object 2 under 2 mm collimation; (a) Simple threshold method; (b) the SST.

Two limitations of the SST algorithm are apparent. The first is caused when identifying Gaussian curve parameters. Figure 3-10 shows frequency distribution curves from four corresponding sampling regions, A to D in Object 1. Pure components, solid and void, can be properly represented by two well-defined Gaussian distribution curves. However, if the void is water filled as in Figure 3-11, the estimation of Gaussian curve parameters becomes more difficult. As the solid phase is reduced, identification of its peak becomes problematic. The second limitation results from the magnitude of the mixed-component voxels. Since the proposed algorithm is based on the Gaussian distribution of pure components by CT scanning, the success of the algorithm is heavily limited by the scanning resolution as shown in Figure 3-12. When compared to 1.5-mm collimation, the 3 mm collimation void component peak has vanished with the increase of mixed-component voxels. The proposed segregation algorithm can identify feature size down to 2-mm in diameter using 1.5-mm collimation size under high attenuation difference. However, the capability of identifying of low-attenuation difference objects is restricted to 5-mm using 1.5-mm collimation size.

Threshold Sensitivity Analysis

A key distinction of the SST and the conventional methods is the way they define and handle the threshold selection. The proposed algorithm applies the threshold only on mixed-component group while the simple threshold method is applied to the entire image domain. I compare volume fractions from void space of test Object 2 under 2-mm

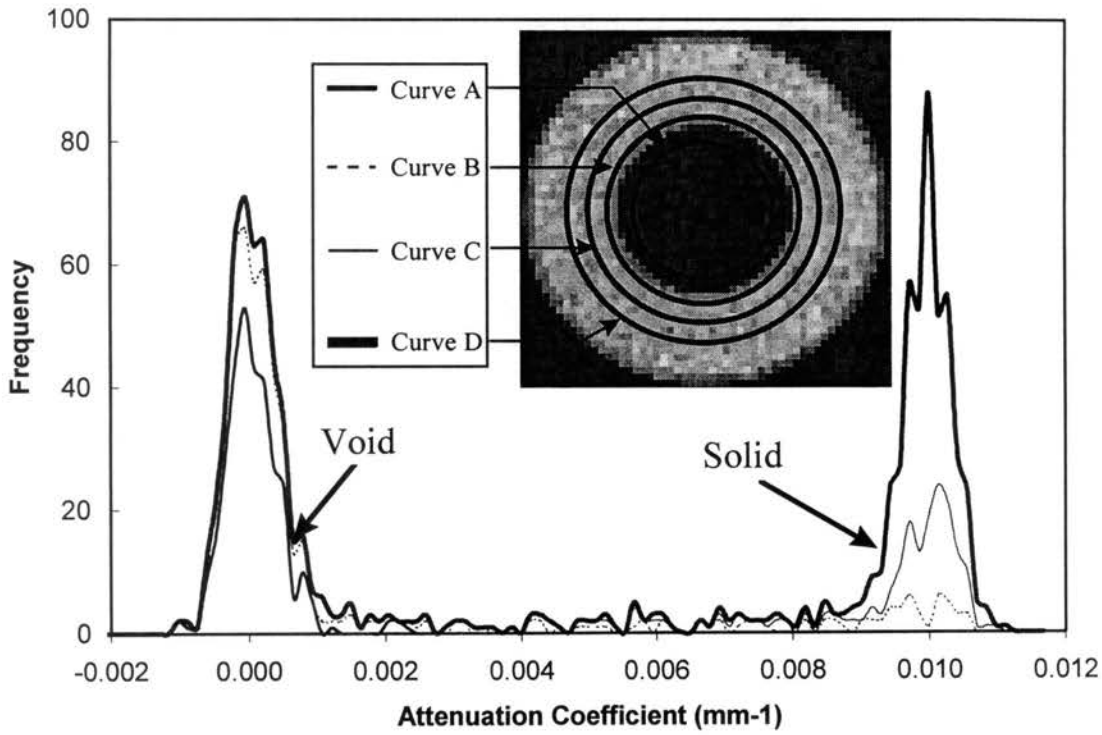


Figure 3-10. Four frequency distribution curves and orresponding sampling regions using Object 1 under 2-mm collimation.

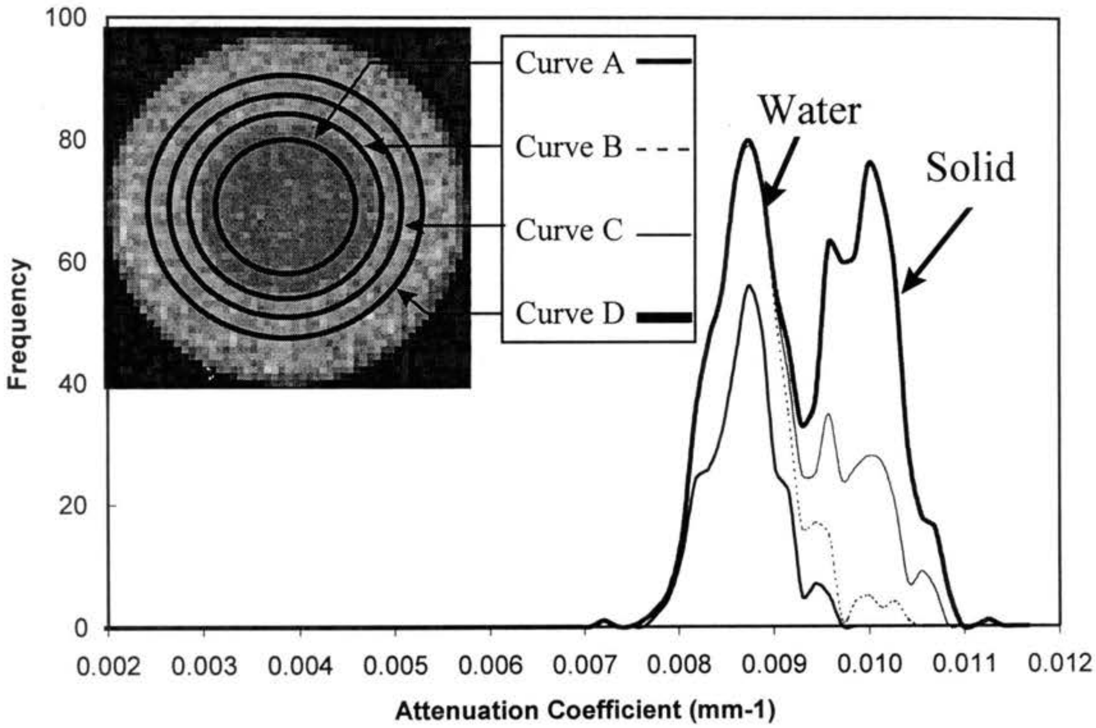


Figure 3-11. Four frequency distribution curves and corresponding sampling regions using Object 1 with water filled void space under 2-mm collimation.

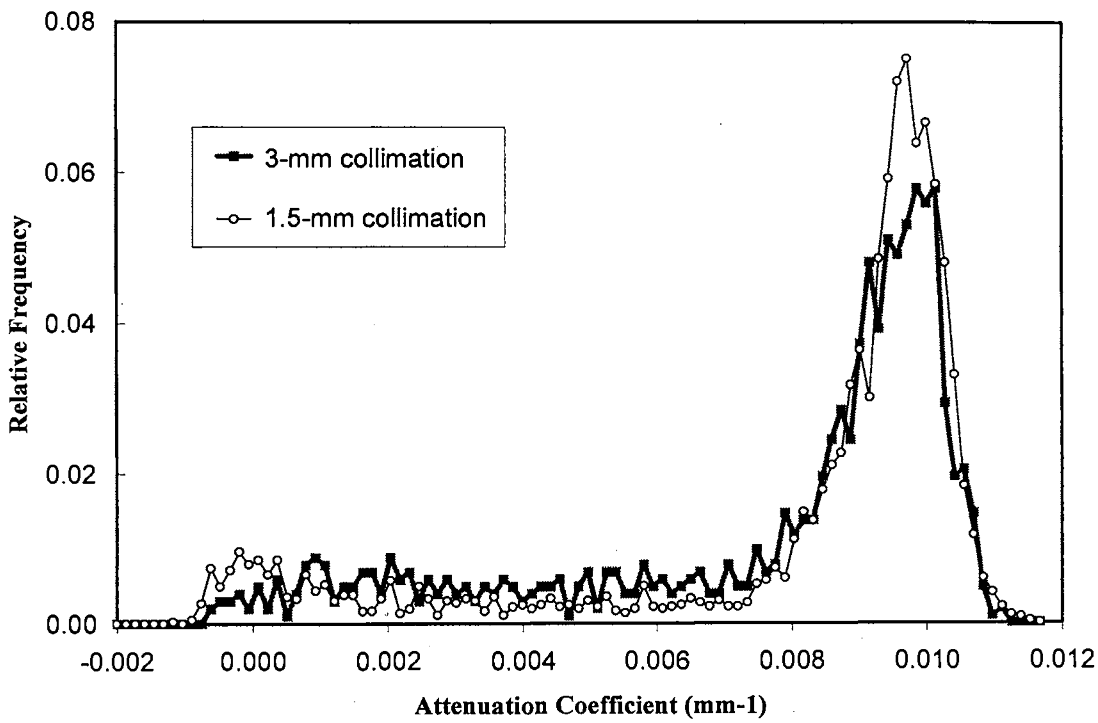


Figure 3-12. Loss of void frequency distribution using multiple scanning resolution on Object 2.

collimation with a relative threshold value for both cases. The relative threshold value for the simple threshold method is defined as the ratio of the selected threshold to the mean threshold value between two peaks, while that for the SST is the ratio between the selected and the previously defined volume fraction of mixed-component voxels by the AFD. Figure 3-13 shows volume fraction of the void voxels varies from 23 to 25 % with the threshold changing from 0.3 to 1.8, while it varies from 25 to 50 % as the simple threshold ratio increase from 0.5 to 1.8. The proposed method is very insensitive to the change of thresholds, comparing to the conventional simple threshold method.

Three important aspects are shown. First, the boundaries between pure components are sharply defined in the SST while mid-attenuation rings are always produced by the simple threshold method. Second, volume fraction and spatial distribution of low-contrast features are better identified by the SST than the simple threshold method. Finally, the identification discrepancy of the two methods increases with sample heterogeneity.

Conclusions

Component segregation and the accuracy of the spatial distribution using CT imaging are complicated by scanning resolution, component density and sample heterogeneity. The proposed SST provides an improved method to transform reconstructed images back to the original component identity.

The strength of the SST algorithm is brought about in three ways. First, the SST selects local threshold values instead of the simple threshold method's global value. Second, the statistical voxel classifications are theoretically sound and are expected to

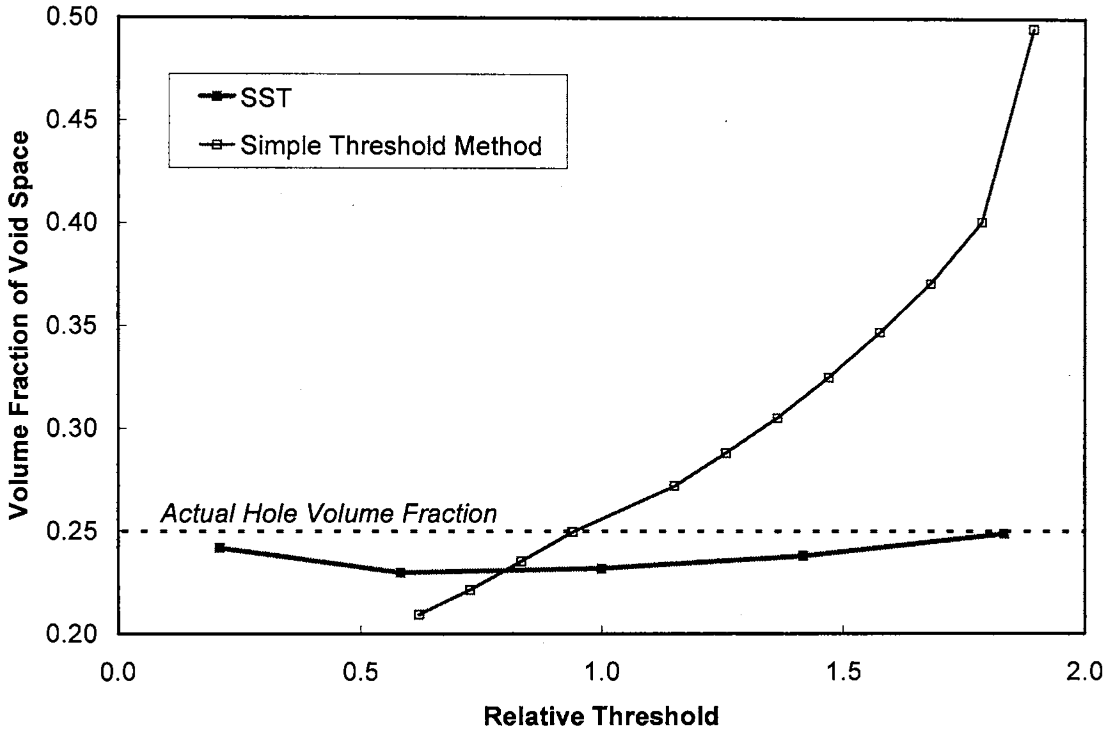


Figure 3-13. Volume fractions of void vs. relative threshold using the SST and simple threshold methods.

extend to x-ray CT images. Third, the class of mixed-component voxels is an indicator for scanning quality and sample heterogeneity.

The images are improved in three ways using SST algorithm,. First, the boundaries between pure components are sharply defined. Mid-attenuation rings seen in the simple threshold method are eliminated. Second, volume fraction and spatial distribution of low-contrast features are better identified. Finally, and possibly most important, identification is less dependent on the increase of sample heterogeneity.

References

Anderson, S. H., R. L. Peyton and C. J. Gantzer, Evaluation of constructed and natural soil macropores using X-ray computed tomography, *Geoderma*, 46, 13-29, 1990.

Anderson, S. H., and J. W. Hopmans (Ed.), *Tomography of Soil-Water-Root Processes,* Soil Sci. Soc. Am. Special Publication No. 36, Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Brown, G. O., Stone, M. L., and J. M. Gazin, Accuracy of gamma-ray computerized tomography in porous media, *Water Resour. Res.*, 29(2), 479-486, 1993.

Gonzales, R. C. and R. E. Woods, *Digital Image Processing,* Addison-Wesley Publishing Company, 1993.

Greves, M. C. J., De Jong, and R. J. St. Arnaud, The characterization of soil macroporosity with CT scanning, *Canadian Journal of Soil Sci.,* 69, 629-637, 1989.

Hopmans, J. W., M. Cislerova and T. Vogel, X-ray tomography of soil properties, in *Tomography of Soil-Water-Root Processes*, edited by S. H. Anderson and J. W. Hopmans, pp. 17-28, SSSA Special Publication No. 36. Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Hsieh, H.T., G.O. Brown, M. L. Stone and D.A. Lucero, measurement of porous media component content and heterogeneity using gamma ray tomography. *Water Resour. Res.*, 1997.(in press)

Kantzas, A, Investigation of physical properties of porous rocks and fluid flow phenomena in porous media using computer assisted tomography. *In Situ.* 14(1):77-132, 1990.

Kantzas, A., Determination of sulfur saturation dolomitic sour gas reservoir using computer assisted tomography. *In Situ*, 15(3):215-246, 1991.

Peyton, R. L., S. H. Anderson and C. J. Gantzer, Measurement of soil structure, water movement and solute transport using computed tomography. *Tech Rep to U.S. Geol. Surv.*, Reston, VA, Grant 14-08-0001-G1643, 77pp, 1992.

Spanne, P., K. W. Jones, L. Prunty, and S. H. Anderson, Potential applications of synchrotron computed microtomography to soil science. in *Tomography of Soil-Water-Root Processes*, edited by S. H. Anderson and J. W. Hopmans, pp. 43-58, SSSA Special Publication No. 36. Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Warner, G. S., J. L. Nieber, I. D. Moore and R. A. Geise, Characterizing macropores in soil by computed tomography, *Soil Sci. Soc. Am. J.*, 53(3), 653-660, 1989.

Warner, G. S. and J. L. Nieber, Macropore distribution in tilled vs. grass-surfaced cores as determined by computed tomography, in *Preferential Flow,* edited by T. J. Gish and A. Shirmohammadi, pp. 192-201, Am. Soc. Agricultural Engineers, St. Joseph, Michigan, 1991.

Chapter IV

QUANTIFICATION OF THE REPRESENTATIVE ELEMENTARY VOLUME OF HETEROGENEROUS POROUS MEDIA USING GAMMA RAY TOMOGRAPHY

Abstract

Two new quantitative procedures were applied to computerized tomography images of a complex dolomite to provide macropore distribution and size estimates of a Representative Elementary Volume (REV). Spatial distribution of both minerals and macropores may be determined to a higher degree of precision than simple thresholding with the use of the attenuation frequency deconvolution (AFD) and the statistical segregation threshold (SST). Those procedures are described in the previous two chapters of this dissertation. Because images resulting from AFD and CSA show better-defined pure component regions and smoother boundaries, the REV can then be determined by integration of the small-scale density and macroporosity over larger and larger volumes. These procedures were applied to two samples of the Culebra Dolomite Member of the Rustler Formation collected at the Waste Isolation Pilot Plant near Carlsbad, New Mexico. Sizes of sample volumes spanned over six orders of magnitude,

from 0.25 to 1 x 10^6 mm^3. While density and macroporosity showed convergence to single values, statistical tests indicated the biggest sample volumes were not sufficiently large for a REV.

Introduction

Transport modeling in porous media is usually based on a continuum model which requires the selection, or an assumption of a Representative Elementary Volume (REV) [*Hubbert,* 1956; *Bear*, 1972], which is defined as the smallest volume for which all averaged geometrical characteristics are single valued functions of the location of that point and time. *Baveye and Sposito* [1984] observed that while intuitively appealing, no known data has been presented to quantify the REV. In a traditional sense, the REV represents the transition from the microscopic deterministic processes of traditional fluid mechanics to the macroscopic processes of porous media flow. However, in practice, the concept has also been applied to characterize both non-homogeneous porous media and large scale properties in fractured media [*Bear*, 1993]. In these cases, separation of REV's can be defined for both matrix and fractures. Along similar lines, heterogeneity has been used to explain porous media with small scale structures such as sedimentary bedding planes [*Corey*, 1977], which exhibit anisotropic transport parameters. Again while intuitive, no data has been presented to show that the structures do in fact possess the properties proposed.

CT images have been intensively used for quantifying macropores and fractures in porous media [*Anderson et al.*, 1990; *Hopmans et al.,* 1994; *Warner et al.*, 1989; *Warner and Nieber*, 1991; *Greves et al.*, 1989]. Followed by the development of the attenuation frequency deconvolution (AFD) algorithm and the statistical segregation threshold algorithm (SST), this study extends such application to gamma CT images of a complex dolomite that contains gypsum, and several porosity types. The performance of the SST in identifying voids will then be compared to a simple threshold method. Finally those results will be used to provide new insight into two closely related problems that have plagued porous media researches, the identification of the REV and its size.

Theory

Statistical Segregation Threshold (SST) Algorithm

With the AFD, several pure component- and one mixed-component-phases within CT images can be defined. In simple terms, the AFD determines bulk volume contents of both phases by fitting parameters in a theoretical distribution to the measured attenuation frequency distribution. The SST then uses the AFD information to first define the location of mixed-component volume elements (voxels) which lay between pure component boundaries using Sobel edge detection which has been proved to be superior in the previous chapter. Edge detection extracts voxels having high attenuation gradients across its neighbors. Voxels having a high probability of being a pure component (peak voxels), are then determined. Those voxels that exclude mixed-component and peak voxels are defined as tail voxels. Finally the nearest neighborhood comparisons is applied to decide the component contents of mixed-component and tail voxel s. The mathematical foundation of the method is briefly presented below.

A measured distribution function, $h(\mu)$, of the attenuation coefficient, μ, is the convolution result of the Dirac delta function, $\delta(\mu)$, used to define pure component attenuation, with the Gaussian distribution, $g(\mu)$, [*Kak and Slaney*, 1988]

$$h(\mu) = \delta(\mu) * g(\mu) \tag{4-1}$$

where $*$ indicates the convolution of δ with g.

When testing real rock and soil, two scanning complications are expected. First, rocks typically have more than one component, including multiple minerals and pore space. Second, when two or more components are interlaced with one another, a large number of voxels has component boundaries crossing through them. Those voxels will have a true attenuation between the density of the pure components. The true attenuation, μ_m^* of a "mixed-component" voxel at position (x, y) made up of K components will be given by

$$\mu_m^*(x,y) = \sum_{k=1}^{K} r_k(x,y)\mu_k^* \tag{4-2}$$

where $r_k(x, y)$ is the volume content of the component k within the individual voxel and K is the total number of components. Therefore, for a CT scanned sample with multiple pure components and a mixed-component phase the measured density frequency is given as

$$h(\mu) = \sum_{k=1}^{K} R_k(\mu) f_k(\mu) + R_m(\mu) f_m^*(\mu') \tag{4-3}$$

where R_k is the fractional volume content of the pure component k and R_m is the volume content of the mixed-component phase. The distribution function $f_k(\mu)$ denotes pure components and $f_m{}^*(\mu')$ is the mixed-component phase. According to the magnitude of the variance in the Gaussian distribution, define voxels into a peak-voxel class, if they fall into the range of one standard deviation to the both side of the pure component mean. Equation 4-3 can be rewritten as

$$h(\mu) = \sum_{k=1}^{K} [R_k^H(\mu) f_k^H(\mu) + R_k^L(\mu) f_k^L(\mu)] + R_m(\mu) f_m^*(\mu') \tag{4-4}$$

where R_k^H and R_k^L are the fractional volume content of the pure component k within tail-voxel and peak-voxel classes, respectively. The distribution functions, $f_k^H(\mu)$ and $f_k^L(\mu)$, denote pure component within tail- and peak-voxel groups, respectively. A conceptualization of this voxel classification is shown in Figure 4-1. In it two pure components bracket a mixed-component phase and both peak and tail voxels and pure component regions are shown.

Voxels are first segregated into pure and mixed-component voxel classes using Sobel edge operation. The pure component class is further divided into peak- and tail-voxel subclasses. Since peak voxels have the highest possibility to be identified as pure components, the component segregation is first applied on those voxels. For voxels within both tail-voxel and mixed-component classes, the identification of pure components uses the nearest neighborhood technique in an iterative fashion. The nearest neighborhood technique first compares and calculates the component fraction of surrounding voxels and then assigns the most dominant component identity to the center voxel.

Representative Elementary Volume (REV)

The representative elementary volume of a statistically homogeneous porous medium is defined as the volume ranges for which all averaged geometrical

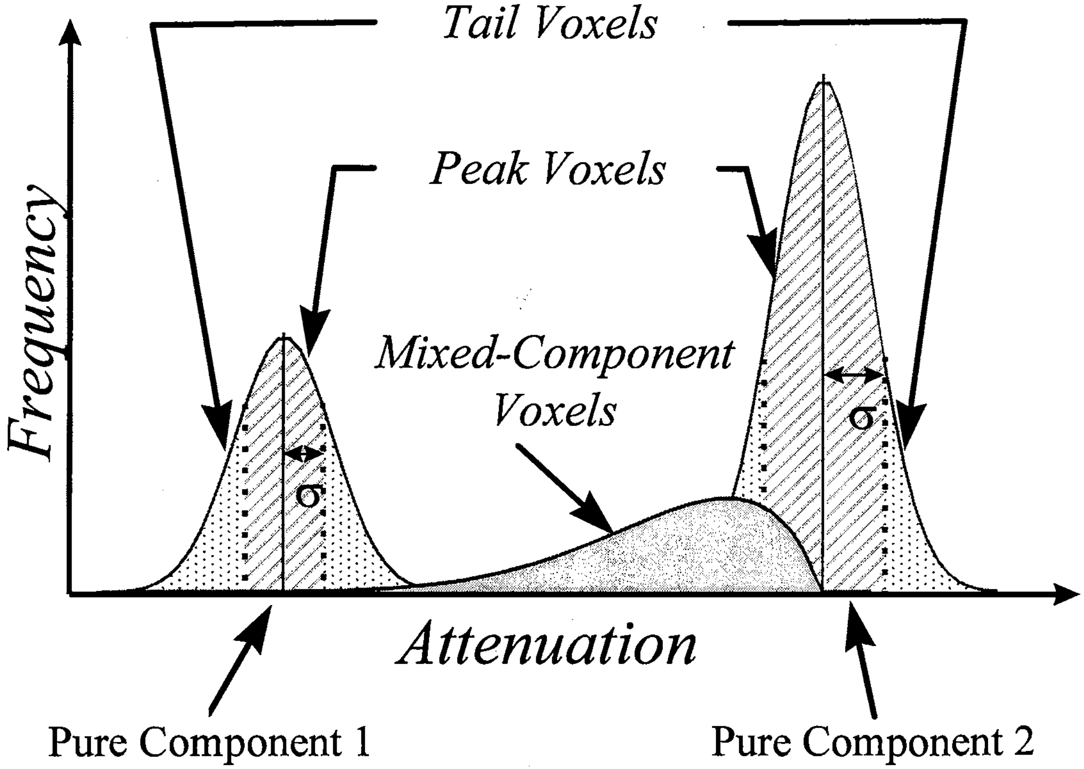


Figure 4-1. Conceptualized voxel classification.

characteristics are single valued functions of the location of that point and time only. Knowledge of the REV is essential for any experiment that regards a porous medium as a continuum. According to *Bear et al.* [1990], a given domain R with a length l is within the range of a REV, if l is bounded by distances l_{max} and l_{min} that represent the upper and lower limits of a REV. It can be written as,

$$l_{min} << l << l_{max} \tag{4-5}$$

A conceptual plot in Figure 4-2 with volume shows that heterogeneity has to be considered for volume scale smaller than U_{min}. That is, no single physical property can be defined to represent the averaged macroscopic quantity at this scale. While the characteristic volume is located between U_{min} and U_{max}, a region of a REV can be found, that satisfies,

$$\left.\frac{\partial \gamma(X,U)}{\partial U}\right|_{U=U_0} = 0 \tag{4-6}$$

where U_0 is a volume of the REV and X is the location coordinate of either 1-D or multiple dimension within the sampling domain R. $\gamma(X,U)$ is the physical property value centered at location X with a volume size of U within a given domain R. The volume size above U_{max} includes more geological structures and the physical properties of the porous media may drift to new values.

It is impractical or impossible to observe all samples within R. A statistical test of the REV size has been derived by sampling the measurable hydrological characteristics [*Bear and Bachmat*, 1993]. A random function, $\gamma(X)$ is regarded as a characteristic function at any point X with volume U. By repeated sampling, additional realizations,

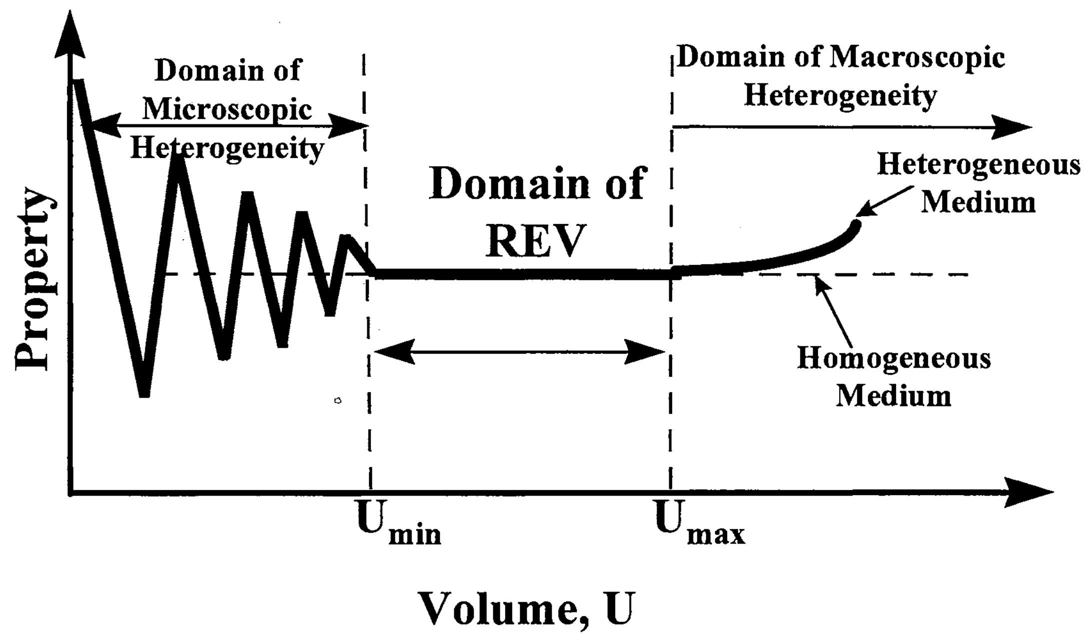


Figure 4-2. Microscopic and macroscopic domains and the representative elementary volume.

$\gamma^{(i)}(X)$, are obtained. $\gamma(X)$ can be treated as a stationary random function in R if

$$E[\gamma(X)] = \theta = constant \tag{4-7}$$

where $E[\gamma(X)]$ is the expected value of γ. The variance at location X is expressed as,

$$\mathrm{Var}[\gamma] = \mathrm{E}\{[\gamma(\mathrm{X}) - \theta]^2\} = \mathrm{constant} \tag{4-8}$$

A domain, R, for which Eqs. 4-7 and 4-8 hold, is referred to as macroscopically homogeneous regarding the property, $\gamma(X)$.

Materials and Methods

Sample

Two dolomitic cores from the Culebra Dolomite member of the Rustler Formation are selected for the study [*Lucero et al.*, 1994]. The VPX-25-9 core is 145 mm in diameter and 100 mm in length, while VPX-26-C1AV is 38 mm in diameter and 52 mm in length. Total core volumes are 1,650,000 and 60,000 mm^3 for VPX-25-9 and VPX-26-C1AV, respectively. They were collected by horizontal drilling at a depth of 218 m in the air intake shaft of the US Department of Energy Waste Isolation Pilot Plant located near Carlsbad, New Mexico. Examination by scanning electron microscope of a separate sample from the same level and location found dolomite and gypsum, and trace amounts of corrensite, quartz and halite. Visual examination of the core confirmed that dolomite and gypsum were the only significant mineral components. Both cores were relatively solid and intact, but demonstrated the fractures, gypsum infilling and vugs typical of WIPP Rustler cores. VPX-25-9 showed considerable gypsum, while VPX-26-C1AV was almost entirely dolomite.

CT Images

The pencil-beam, gamma ray CT scanner of *Brown et al.* [1993] was used here. The monochromatic nature of gamma ray transmission combined with monochromatic detection techniques allows voxel densities to be determined with bounded error. VPX-25-9 was scanned by 120 projects with 120 rays each, and live detector times of 5 seconds, while VPX-26-C1AV were scanned with 90 by 90 array and live detector times of 10 seconds. Thirty-one planes at 3 mm spacing along the axis of VPX-25-9 were collected with 3 mm collimators and 1.5 mm ray spacing. For VPX-26-C1AV, 53 planes at 1 mm axis spacing were scanned with 1.5 mm collimation and 0.5 mm ray spacing. All scans for both cores were reconstructed into a 120 x 120 image array. Image voxel volume produced was 0.14 mm^3 in VPX-26-C1AV and 6.75 mm^3 for VPX-25-9. The cores and their respective scanning resolution were selected to provide the best information possible over the largest scale range possible with the instrument used. Scanning the larger sample with the smaller collimator was infeasible in any practical time.

Hydraulic Properties

Computed Bulk Density

VPX-26-C1AV was air dried when scanned. Due to restrictions for its latter application of the cores, VPX-25-9 was freely but not fully drained. Therefore, the calculated CT attenuation is referred as a "computed" attenuation. *Luo and Wells* [1992] have shown that mass attenuation coefficients are insensitive to mineral composition at the gamma energy used here. *Kelley and Saulnier* [1990] performed extensive measurements on 25 50 mm-diameter Culebra core samples, and found the median dolomite grain density to be 2.83 Mg/m^3 with an average total porosity, including fractures of 0.13. Following the previous discussion, the system is calibrated with gypsum's density of 2.32 Mg/m^3 [*Weast*, 1988]. The analysis of images provided a gypsum attenuation of 0.0171 mm^{-1}. For ease of comparison, all data were converted from attenuation to density by multiplication with the calibration factor, C = 135.7 mm-Mg/m^3. It follows that the computed bulk density, ρ is transformed by

$$\rho(x, y) = C\,\mu(x, y) \tag{4-9}$$

Macroporosity Index

Conventionally defined under hand specimen macropores are visible voids or fractures in porous media. However, because of limited CT scanning resolution, the macroporosity may not well identified from scanned images. Therefore, each voxel at position *(x, y)* must be interpreted as a mixture of solid, micropore and macropore components. A computed bulk density may be found as

$$\rho(x,y) = \rho_s'(x,y)[1 - r_{macro}(x,y)] \tag{4-10}$$

where $\rho_s'(x, y)$ becomes the density of solid component with micropores and r_{macro} is the volume contents of macropore. Therefore, in this study, a characterized "macroporosity index" voxel can be interpreted as that has a high portion of macropores and low solid components. The volume fraction of the index, ϕ_m, is defined as

$$\phi_m = \frac{N_m}{N_T} \quad (4\text{-}11)$$

where N_m is the number of the macroporosity index voxels and N_T is the total number of voxels. This index is a measure of the frequency of macropores, and not their size. Thus, both voxels completely void of material and those with very small voids are included.

REV Sampling Procedure

Computed bulk density and Macroporosity Index are selected for analyzing the REV concept. An undisturbed region was selected from the original cores. The defined sample domain for VPX-25-9 core was 102 x 102 x 93 mm^3, or 0.97 liters in volume and that for VPX-26 core is 27 x 27 x 52 mm^3, 0.038 liters. Various sample volumes for the REV analysis were obtained by averaging voxel values from continuously expanding rectangular prisms. The continuously expanding procedure collects eight curves by continuously expanding sampling volume size from eight corners as shown in Figure 4-3. Each sample volume incorporated the former. While expansion continues, each cube increases two voxels on a side and one in height for VPX-25 core, and one on all 3-D directions for VPX-26 core. At the maximum, each of them will fill the entire sample domain. The sampled volumes are overlapped after the sampling length exceeds one-half of the domain length.

Statistical Hypothesis Test

Upon meeting Eq. 4-7 and 4-8, the tested sample size can be accepted as a representative volume. That is, the mean values of a hydrological property under the

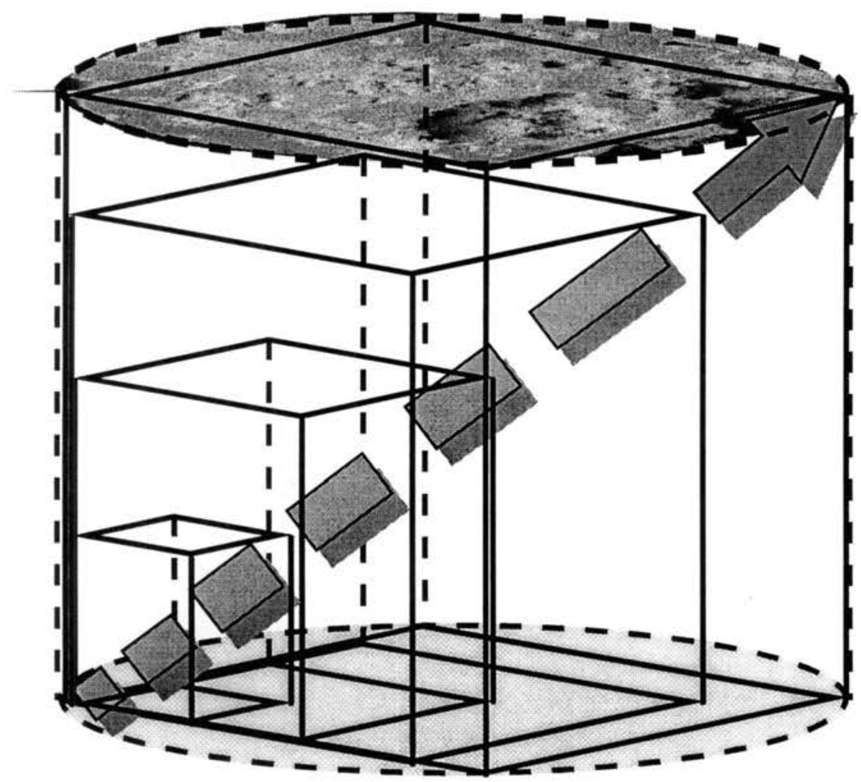


Figure 4-3. Schematic of continuously expanding sampling procedure. Data is collected by increasing sampling volume from one corner to its diagonal.

same volume size at different locations have statistically indifference. According to *Devore* [1995], a single-factor ANOVA is selected to test more than two populations or treatment means. The test objective is to determine whether the equal-volume rock masses that were sampled from different locations of the same sample core possess a statistically indifferent physical property. The testing hypotheses is described as,

$$H_0: \mu_1 = \mu_2 = \ldots = \mu_n \quad (4\text{-}12a)$$

$$H_a: \textit{at least two of the } \mu_n\textit{'s are different} \quad (4\text{-}12b)$$

where μ_n, is the mean value at location n of volume size U. Since a high degree of variation is seen among the smaller volume samples, they were visually compared and their possibility of passing the statistical test rejected. Only those samples with volume size larger than 400 cm^3 were used to perform the test.

By gradually increasing the sampling volume size, a volume size within which all samples have a statistically indifferent physical property is determined. The lower volume limit of the REV, l_{min} is found if the selected property under such volume passes the hypothesis test defined above. Through increasing sample volume size, tests should always fail to reject the hypothesis until reaching the upper volume limit of the REV, l_{max}, where the hypotheses test of indifference of property is being rejected. A domain between these two points is referred to as macroscopically homogeneous with respect to the investigated physical property.

Results and Discussions

Component Segregation Results

According to the previous chapter, the strength of the SST over a simple thresholding method has been demonstrated on four test objects. Performance of the algorithm on real samples is demonstrated here on four planes, 4, 14, 21 and 30, in VPX-25-9. The raw CT image of the central 68 x 68 voxel region of each are shown in Figure 4-4. Figure 4-5 presents the final ADF and SST results for each section. As visually inspected in the raw images and the deconvoluted Gaussian distributions, the sections show considerable variation.

Table 4-1 shows the component fraction results for VPX-25-9 using SST algorithm and simple threshold method. The volume fraction of dolomite defined by two methods has a discrepancy of less than 9 %. However, defined volume fraction of macroporosity ranges between 6 to 20 % by SST algorithm and 4 to 11 % by simple threshold method, with a up to 20 % discrepancy. Figure 4-6 of the component images from SST algorithm show the better-defined component regions, smoother component boundaries and less inclusions inside the components. The mid-density boundaries as shown in Figure 4-7 by simple threshold process indicate the incapability of treating the mixed-component voxels by the simple threshold method. Higher percentages of mixed-component phase were found in Planes 21 and 30 than other planes and indicates higher

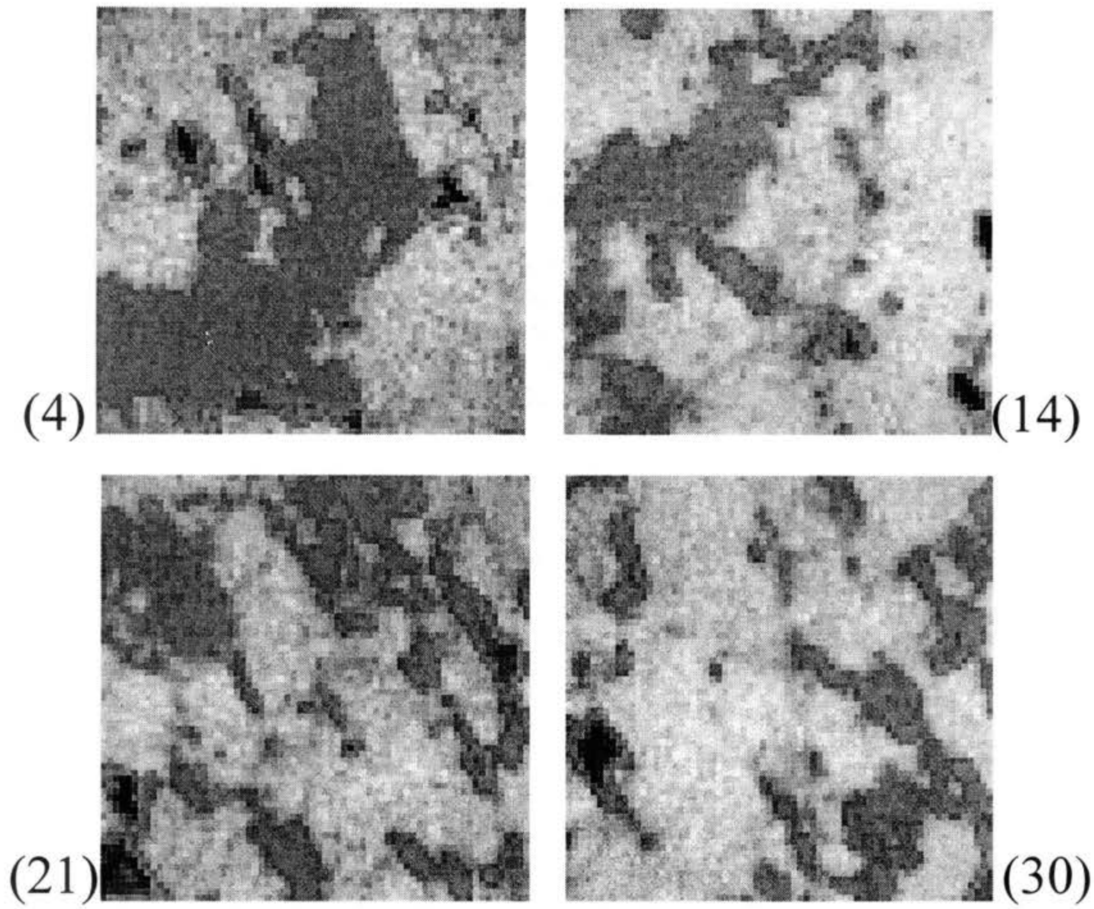


Figure 4-4. Four 68x68 CT images, cropped from 120x120 image arrays, used for component segregation algorithm from Planes 4, 14, 21 and 30 for VPX-25-9.

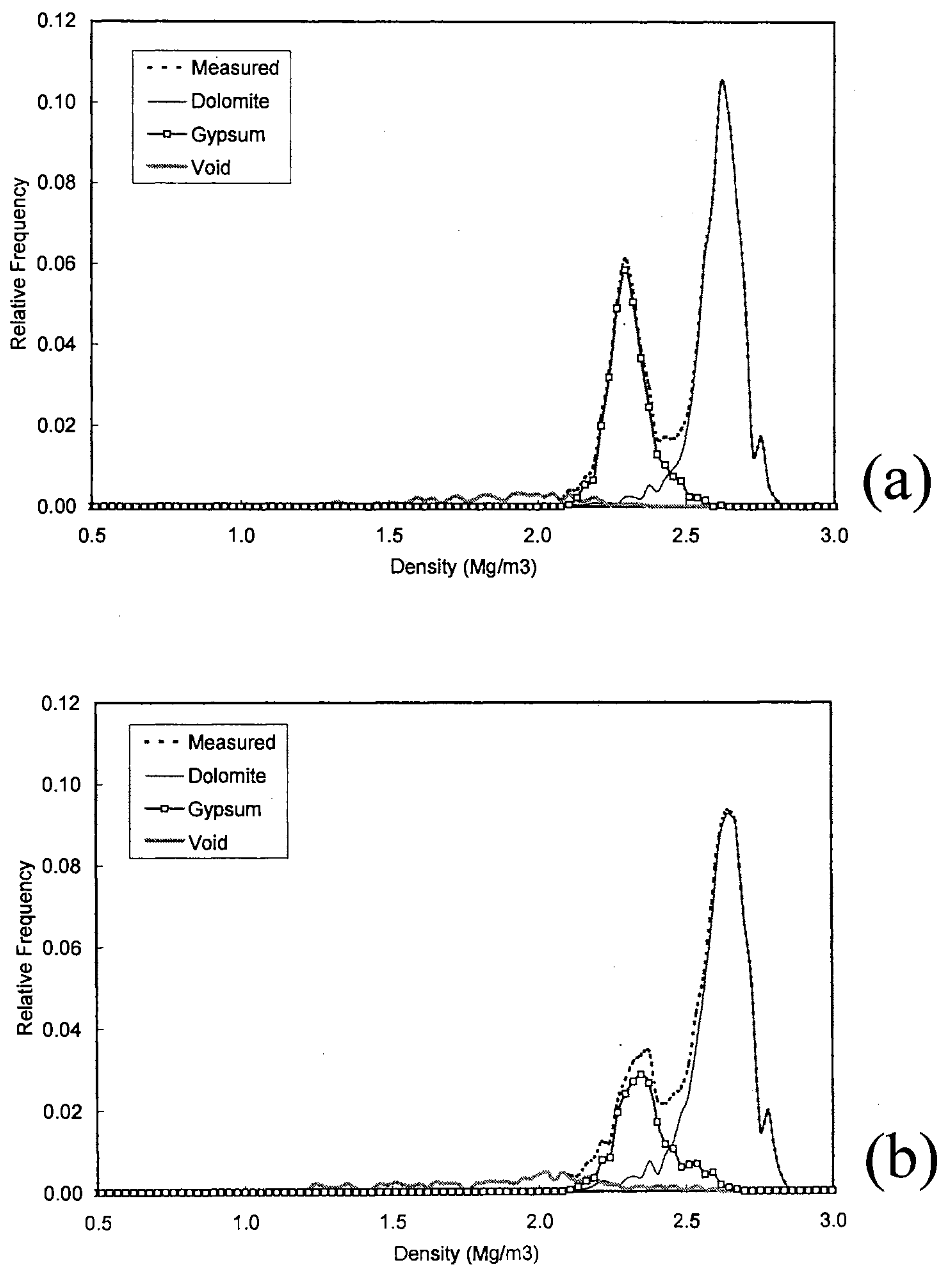
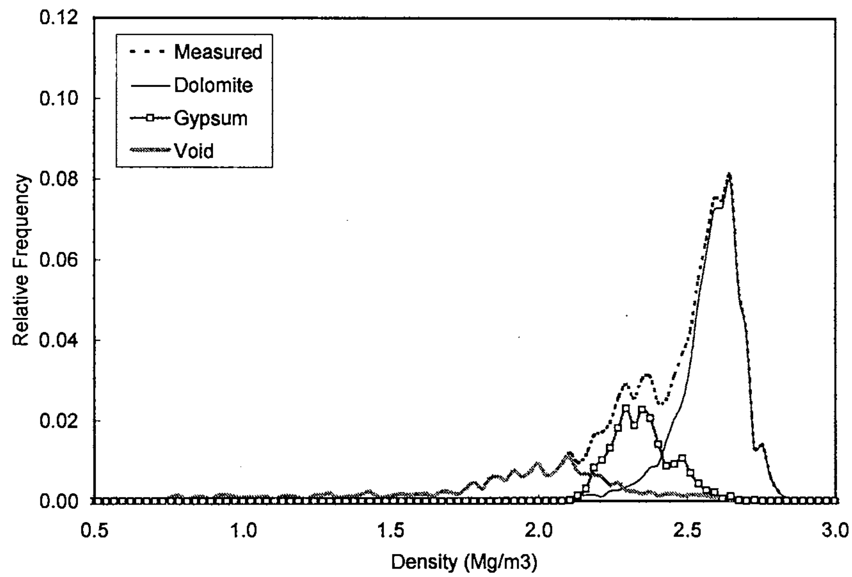
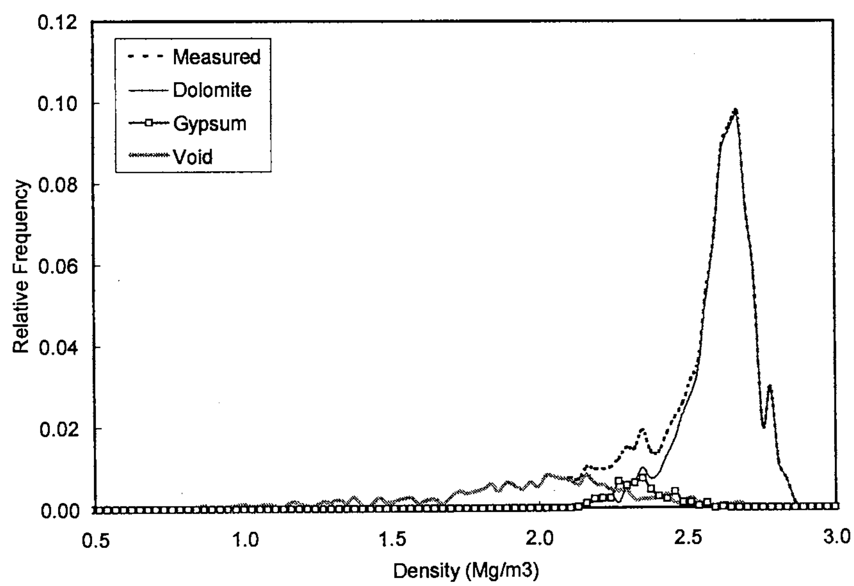


Figure 4-5. The deconvoluted component histograms for (a) Plane 4, (b) Plane 14, (c) Plane 21, (d) Plane 30 with 68x68 voxels from VPX-25-9.



(c)



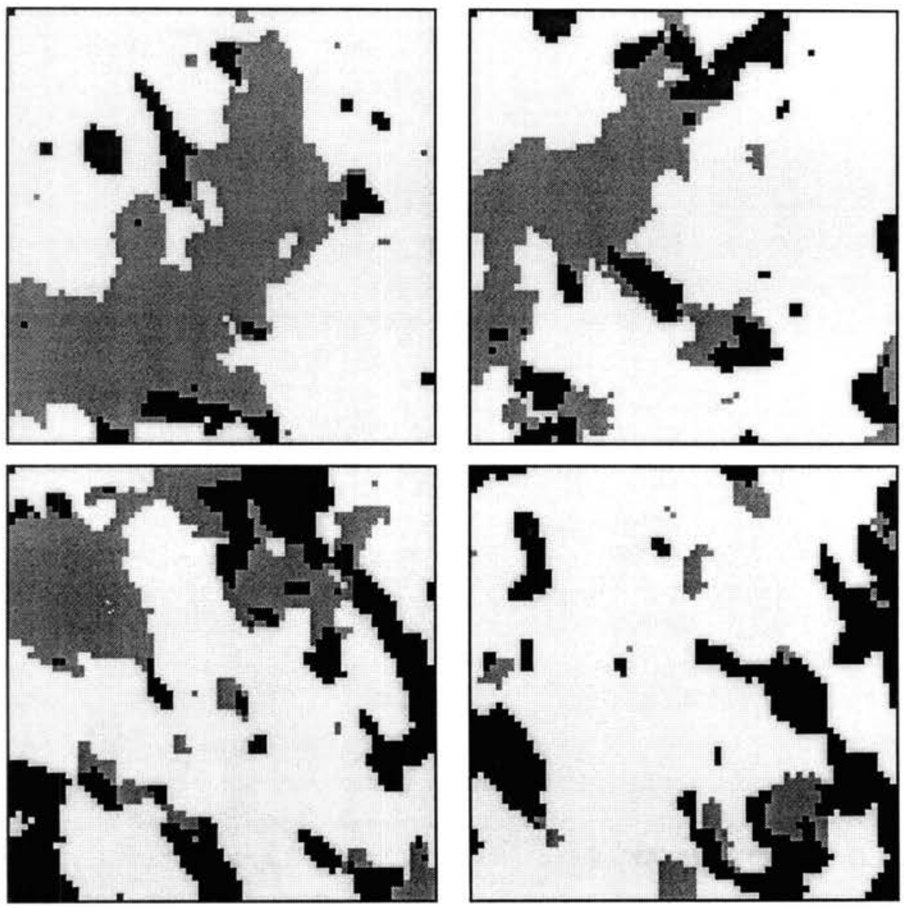
(d)

Figure 4-5. (contd.) The deconvoluted component histograms for (a) Plane 4, (b) Plane 14, (c) Plane 21, (d) Plane 30 with 68x68 voxels from VPX-25-9.

Table 4-1. Results of the AFD, SST algorithm and simple threshold method.

	AFD			SST			Simple Threshold Method		
Sample	dol	gyp	mixed	dol	gyp	mi	dol	gyp	mi
Plane 4	0.59	0.25	0.17	0.61	0.33	0.06	0.60	0.36	0.04
Plane 14	0.52	0.16	0.32	0.68	0.22	0.10	0.65	0.28	0.07
Plane 21	0.52	0.11	0.37	0.60	0.20	0.20	0.59	0.30	0.11
Plane 30	0.58	0.02	0.40	0.76	0.05	0.19	0.69	0.20	0.11

Note: dol- dolomite; gyp- gypsum; mixed- mixed-component voxels; mi- macroporosity index



(4) (14) (21) (30)

Figure 4-6. Component segregated images generated by the proposed algorithm for Planes 4, 14, 21 and 30 from VPX-25-9.

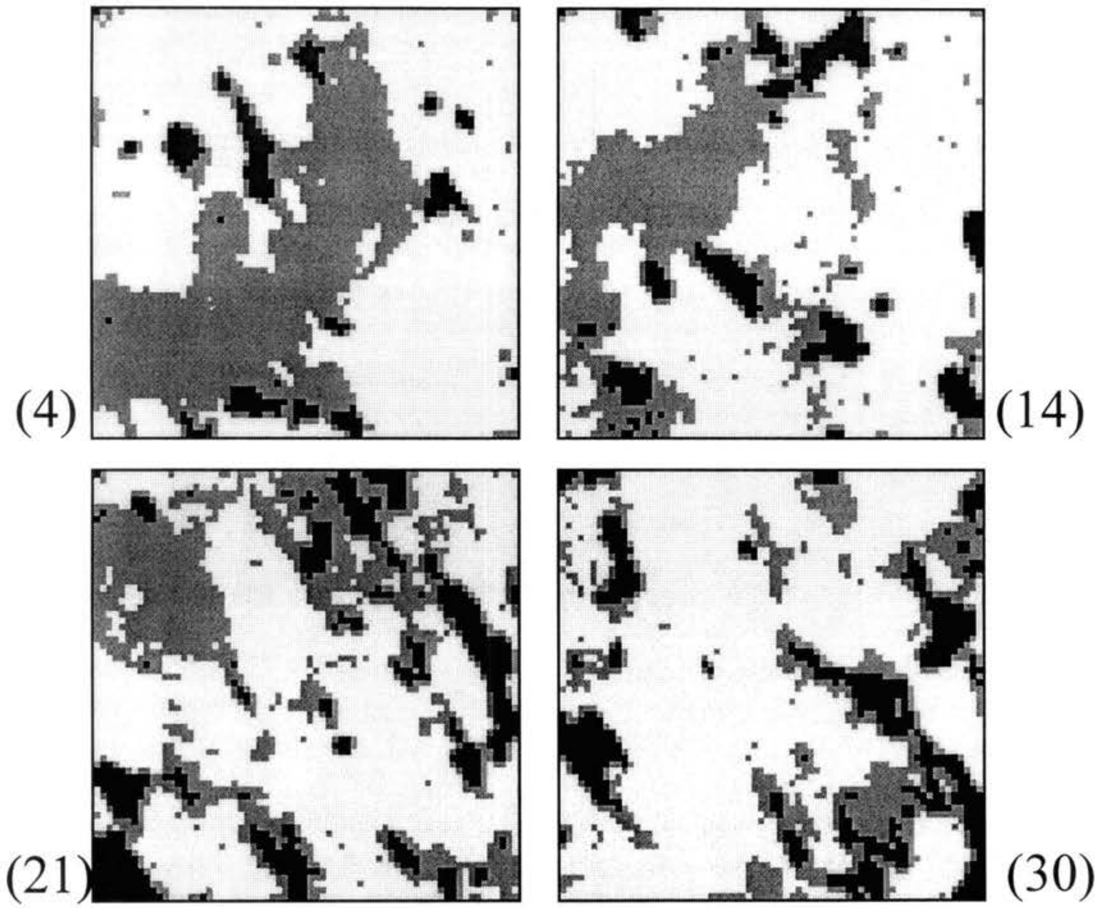


Figure 4-7. Component images generated by the conventional simple threshold method for Planes 4, 14, 21 and 30 from VPX-25-9.

sample heterogeneity, which may be the cause for the increase of macropore discrepancy between the two component separation operations.

Computed Bulk Density and Macroporosity Index

Figure 4-8 shows the distribution of computed bulk density and macroporosity index of 31 planes along VPX-25-9 core length. Heterogeneity is clearly seen through the core while macroporosity index varies between 5 and 18%, while the corresponding computed bulk density changes between 2.42 and 2.54 Mg/m^3. Volume fraction of macropore varies from 5 to 35 % for VPX-26-C1AV and that of density from 2.33 to 2.58 Mg/m^3 as shown in Figure 4-9. Figure 4-10 presents the linear regression relationship between macroporosity index and computed bulk density with *R-squares* of 0.37 and 0.85 for VPX-25-9 and VPX-26-C1AV cores, respectively. The component variation can be postulated by the correlation plot shown in Figure 4-10. As VPX-26-C1AV data set is clustered above VPX-25-9, a larger high density component, or dolomite, is anticipated. Since little gypsum was found in VPX-26-C1AV, the good linear correlation between macroporosity index and computed bulk density actually explains a density variation in a two-component system. Since gypsum content in VPX-25-9 varies along the core, the correlation between bulk density and macroporosity index is reduced and the linear relationship could be expected to disappeared if a third component also fluctuated.

Representative Elementary Volume

Figure 4-11 shows two groups of distribution curves using density and macropore properties for VPX-26-C1AV whose volume size is ranged from 0.14 to 4 $x10^4$ mm^3.

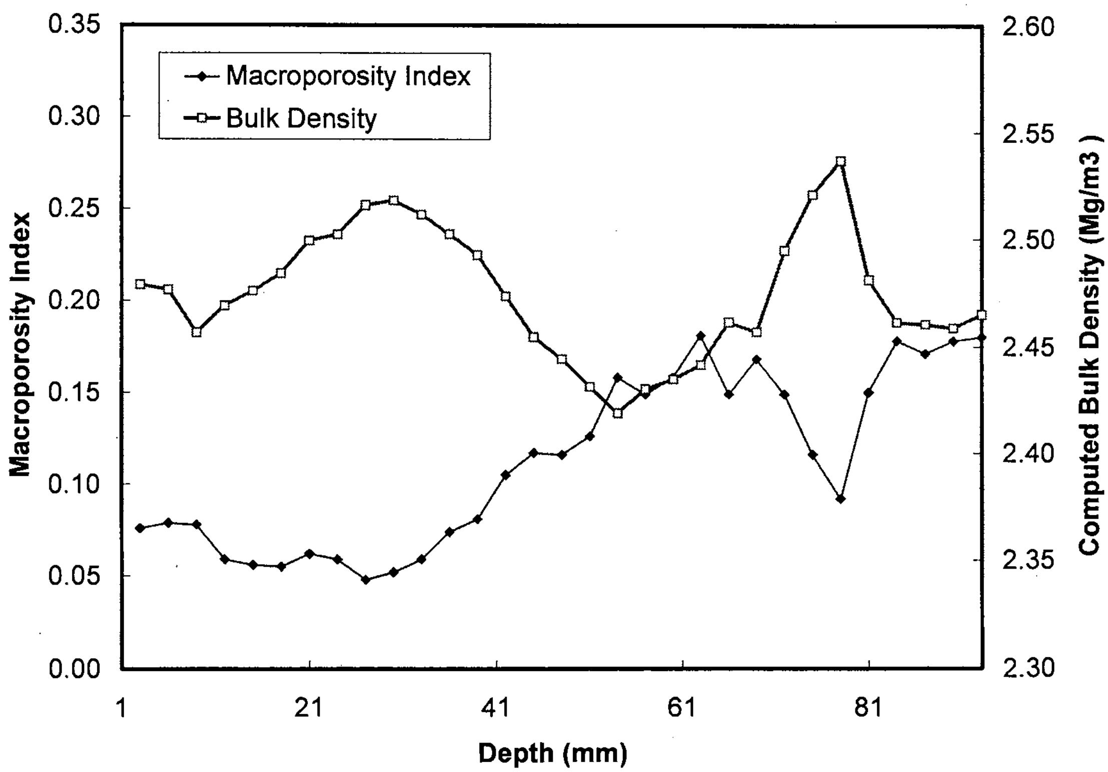


Figure 4-8. Spatial distributions for macroporosity index and computed bulk density along the VPX-25-9 core length.

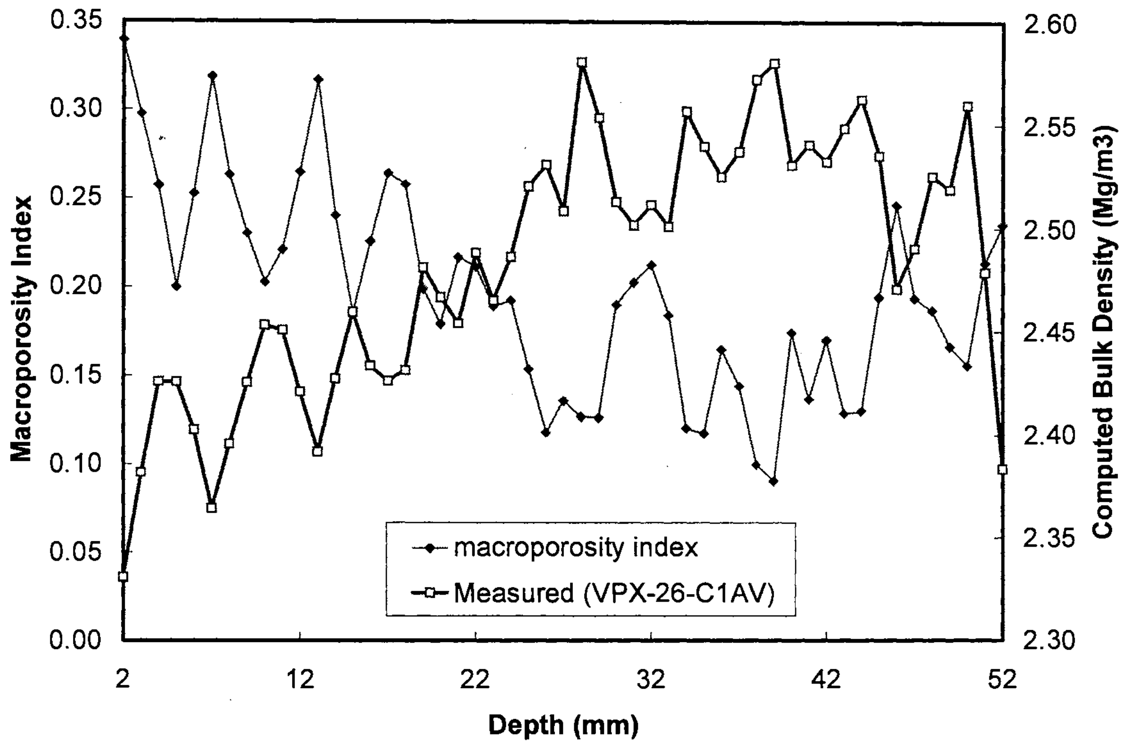


Figure 4-9. Spatial distributions for macroporosity index and computed bulk density along the VPX-26-C1AV core length.

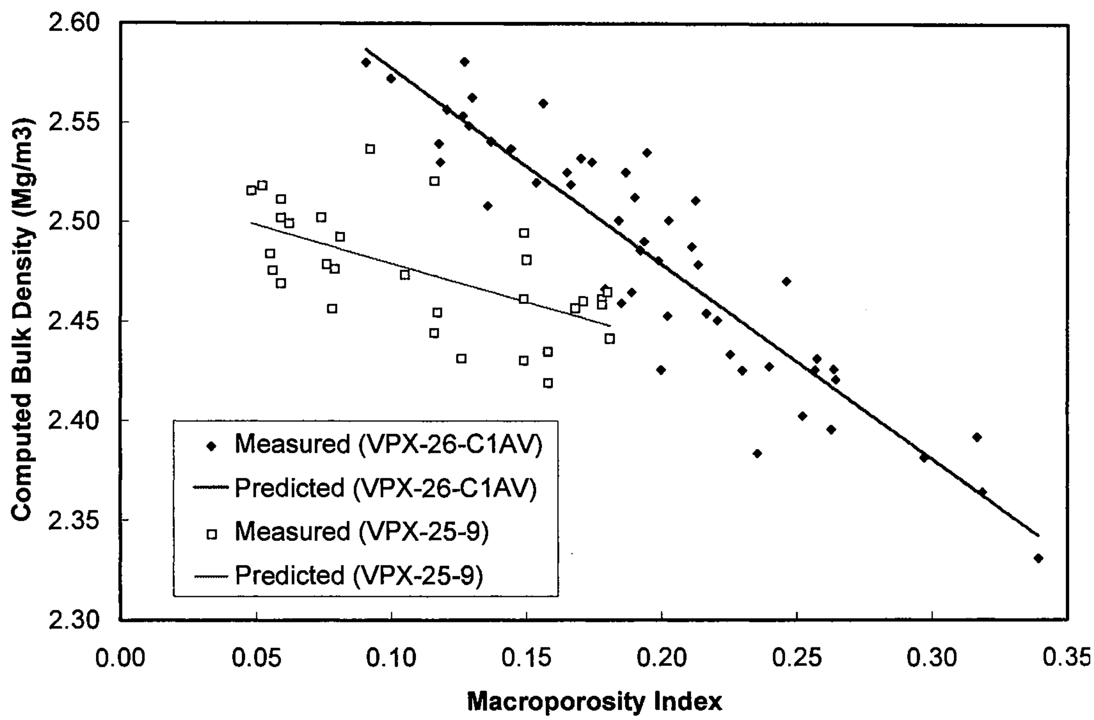


Figure 4-10. Correlation between macroporosity and computed bulk density for both VPX-25-9 and VPX-26-C1AV cores.

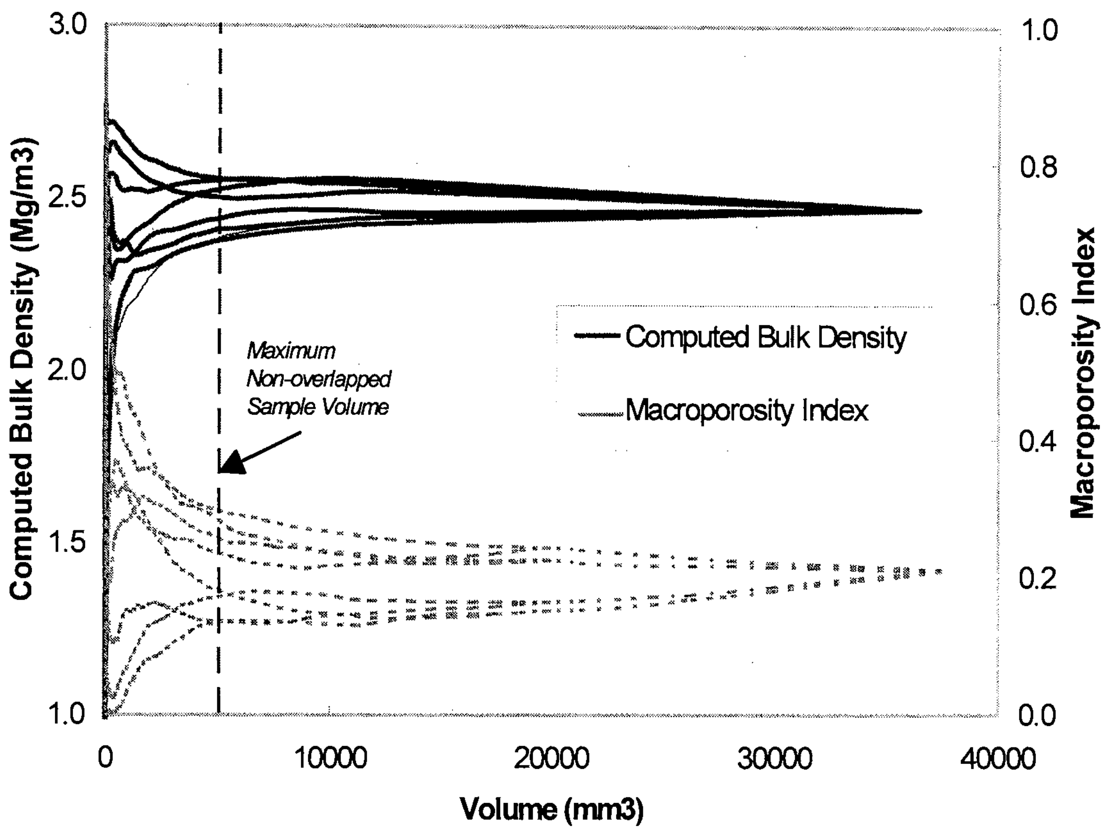


Figure 4-11. The REV distribution patterns for VPX-26-C1AV from both computed bulk density and macroporosity index.

The bulk density oscillates between 0.0 and 2.86 Mg/m^3 and converges to 2.48 Mg/m^3 at its largest volume. Macroporosity index varies between 1 and zero and approaches to 0.21 at last. By expanding volume from eight corners outward, the sampled volumes start to overlap after about 5,000 mm^3 as the line delineated in Figure 4-11. It illustrates the postulated REV curve by *Bear* [1972]. Along the same line, Figure 4-12 shows the same REV pattern from VPX-25-9 core with a 25 times larger volume. While the density change magnitude is similar to that from VPX-26-C1AV, its macroporosity index goes toward 0.11 at the maximum volume.

Statistical Hypothesis Test

The statistical single-factor F-test results for VPX-25 core in Table 4-2 show the density property fails to pass the hypotheses test for volume size below 800,000 mm^3. The second F-test results in Table 4-3 using macroporosity index, indicate the rejection scenarios for all tested volume sizes. As concluded in Tables 4-2 and 4-3, the 800,000 mm^3 is not large enough to be qualified as a representative volume with respect to density for the Culebra dolomite sample examined. However, while the statistical test failed, both Figures 4-11 and 4-12 show that bulk density is approaching a constant value at the full sample size. This indicates that the REV may be at or near the volume tested, but the test used here is too restrictive. Further research is needed to better define the most appropriate statistical test for this data.

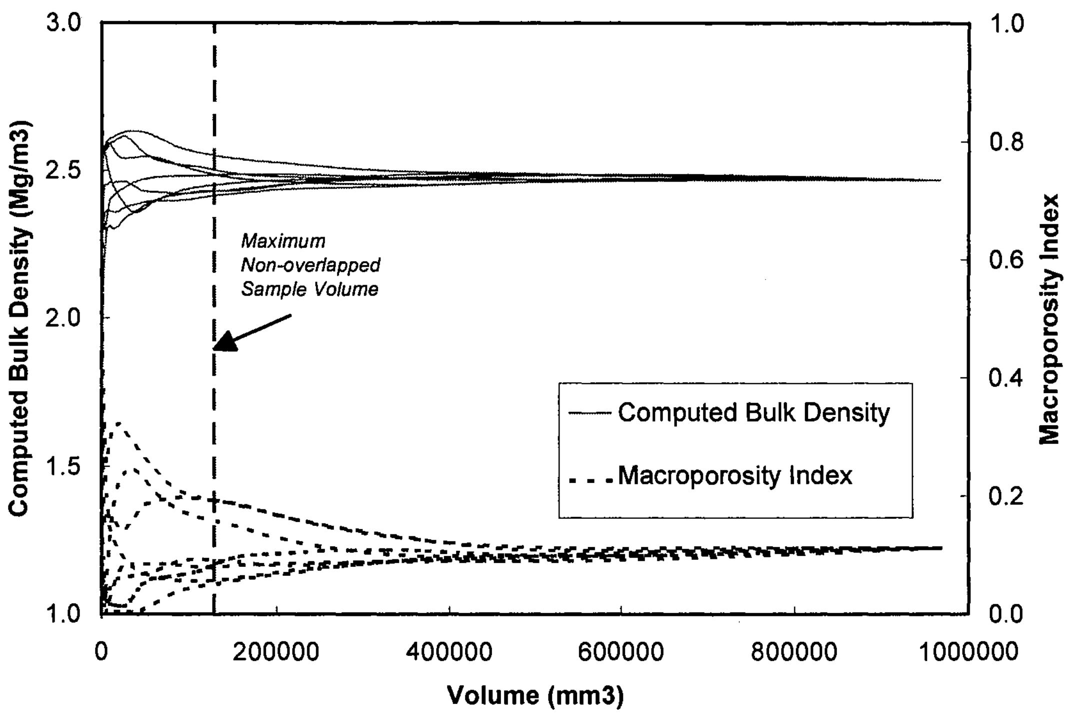


Figure 4-12. The REV distribution patterns for VPX-25-9 from both computed bulk density and macroporosity index.

Table 4-2. Statistical result using one-factor F-test for determining size of the REV by computed bulk density from VPX-25-9 core. α is set to 0.05 for all tests and *df2* is large enough to be set to infinity.

Volume (cm^3)	*df1* *(treatment)*	*df2* *(Error)*	*F value*	$F_{\alpha, df1, df2}$	*Test Result*
421	7	499992	165.9		Reject H_0
475	7	562424	152.3		Reject H_0
531	7	629848	114.1		Reject H_0
593	7	702456	86.0	2.01	Reject H_0
659	7	780440	67.8		Reject H_0
729	7	863992	48.3		Reject H_0
804	7	953304	29.4		Reject H_0

Table 4-3. Statistical result using one-factor F-test for determining size of the REV by macroporosity index from VPX-25-9 core. α is set to 0.05 for all tests and *df2* is large enough to be set to infinity.

Volume (cm^3)	*df1* *(treatment)*	*df2* *(Error)*	*F value*	$F_{\alpha, df1, df2}$	*Test Result*
421	7	499992	81.9		Reject H_0
475	7	562424	57.9		Reject H_0
531	7	629848	45.5		Reject H_0
593	7	702456	34.2	2.01	Reject H_0
659	7	780440	31.5		Reject H_0
729	7	863992	25.0		Reject H_0
804	7	953304	18.2		Reject H_0

Conclusions

Compared to the simple threshold method, the SST algorithm provides a smoother component boundaries and less inclusions within the pure components using real sample cores in this study. The long postulated REV pattern was verified by determining the correlation fluctuation pattern of computed bulk density and macroporosity index versus the change of volume size. The statistical test results used failed to define a REV size for Culebra dolomite with the maximum tested volume size of 800,000 mm^3.

References

Anderson, S.H., R.L. Peyton and C.J. Gantzer, Evaluation of constructed and natural soil macropores using X-ray computed tomography, *Geoderma*, 46, 13-29, 1990.

Baveye, P. and G. Sposito, The operation significance of the continuum hypothesis in the theory of water movement through soils and aquifers, *Water Resour. Res.*, 20, 521-530, 1984.

Bear, J., *Dynamics of Fluids in Porous Media*, American Elsevier, New York, 1972.

Bear, J. and Y. Bachmat, *Introduction to Modeling of Transport Phenomena in Porous Media*, Kluwer Academic Publishers, Dordrecht, 16-28, 553pp, 1990.

Bear, J., Modeling flow and contaminant transport in fractured rocks, In *Flow and Contaminant Transport in Fractured Rocks*, edited by J. Bear, G. DeMarsily, and C.F. Tsang, Academic Press, New York, 1993.

Brown, G. O., M. L. Stone and J. M. Gazin, Accuracy of gamma ray computerized tomography in porous media, *Water Resour. Res.*, 29(2), 479-486, 1993.

Corey, A.T., *Heterogeneous Fluids in Porous Media*, Water Resources Publications, Fort Collins, Colorado, 1977.

Devore, J. L., *Probability and Statistics for Engineering and the Sciences*, 4th Ed., Wadsworth Publishing Company, Belmont, California, 1995.

Greves, M. C. J., De Jong, and R. J. St. Arnaud, The characterization of soil macroporosity with CT scanning, *Can. J. of Soil Sci.*, 69, 629-637, 1989.

Hopmans, J.W., M. Cislerova, and T. Vogel, X-ray tomography of soil properties, in *Tomography of Soil-Water-Root Processes*, edited by S. H. Anderson and J. W. Hopmans, pp. 17-28, SSSA Special Publication No. 36. Am. Soc. Agronomy-Soil Sci. Soc. Am., Madison, Wisconsin, 1994.

Hsieh, H. T., G. O. Brown, M. L. Stone and D. Lucero. Measurement of porous media component content and heterogeneity using gamma ray tomography, *Water Resour. Res.*,1997. (in press)

Hubbert, M. K., Darcy's law and the field equations of the flow of underground fluids, *Trans. Amer. Inst. Min. Met. Eng.*, 207, pp. 222-239, 1956.

Kak, A. C., and M. Slaney, *Principles of Computerized Tomographic Imaging*. IEEE Press, New York, 1988.

Kelley, V. A., and G. J. Saulnier, Jr., Core analysis for selected samples from the Culebra Dolomite at the Waste Isolation Pilot Plant site, *Contractor Report SAND90-7011*, Sandia National Laboratories, Albuquerque, New Mexico, 1990.

Luo, X., and L. G. Wells, Evaluation of gamma ray attenuation for measuring soil bulk density: Part 1. Laboratory Investigation, *Trans. of Am. Soc. Ag. Eng.*, 35(1), 17-26, 1992.

Lucero, D. A., F. Gelbard, Y. K. Behl, and J. A. Romero, Test Plan for Laboratory Column Experiments for Radionuclide Adsorption Studies of the Culebra Dolomite Member of the Rustler Formation at the WIPP site, WIPP test plan, *TP95-03*, Sandia National Laboratories, Albuquerque, New Mexico, 1994.

Peyton, R. L., B. A. Haeffner, S. H. Anderson and C. J. Gantzer, Applying X-ray CT to measure macropore diameters in undisturbed soil cores, *Geoderma*, 53, 329-340, 1992.

Warner, G. S., J. L. Nieber, I. D. Moore and R. A. Geise, Characterizing macropores in soil by computed tomography, *Soil Sci. Soc. Am. J.*, 53(3), 653-660, 1989.

Warner, G.S. and J.L. Nieber, Macropore distribution in tilled vs. grass-surfaced cores as determined by computed tomography, in *Preferential Flow*, edited by T. J. Gish and A. Shirmohammadi, pp. 192-201, Proceedings of the National Symposium, Am. Soc. Agricultural Engineers, St. Joseph, Michigan, 1991.

Weast, R. C. (Ed.), *Handbook of Chemistry and Physics*, pp. F-1, 1st Student Edition, CRC Press, Inc., Boca Raton, FL, 1988.

Chapter V

FUTURE RECOMMENDATIONS

The objective of the study was to develop two quantitative algorithms that help to analyze gamma CT images. The attenuation frequency deconvolution algorithm (AFD) were presented in Chapter II, while the spatial segregation threshold algorithm (SST) for quantifying pure component distribution was shown in Chapter III. In Chapter IV both algorithms have been applied to determine physical properties of porous media and have addressed the fundamental concept of the REV. It is concluded that the proposed algorithms provide a superior tool for quantifying CT images. The objective of developing quantitative CT data analysis has been achieved and its capability also linked to a broader hydrological application.

Further research should focus on integrating quantified CT results with porous media theory. First, preliminary results have shown that scanning resolution and sample heterogeneity impacts the data interpretation. Systematic approaches to quantify the correlation among CT resolutions, feature size and sample heterogeneity are needed. In general scanning resolution of gamma CT is controlled by the collimation size, however, the actual identifiable feature size of core samples using CT is determined by both collimation size and sample heterogeneity. By developing a strategy that optimizes the scanning parameters and feature identification capability, a knowledge baseline for a systematic data analysis procedure that minimizes the interpretation divergence can be provided.

Second, CT's non-destructive planer scanning capability provides a means for a sequential 3-D physical property and fluid transport phenomena's analysis. Since mathematical morphology has been the back bone of the systematic image analysis on anatomy, geology, petrology and other sciences, it is chosen to explore the spatial geometry of porous media presented by CT imaging. While 3-D property geometry and their corresponding fluid distribution profiles, generated by emission gamma CT are identified, the refined correlation between interior structures and transport property can provide a useful knowledge for revising ground water transport models that consider either uniform or non-uniform flow scenarios.

Finally, while gamma ray CT provides an accurate measurement of properties of porous media, its main limit is the scanning speed. Contrarily, X-ray CT provides faster scanning speed at the expense of additional system errors. A transformation of the developed methods from gamma CT to X-ray CT will not only contribute high speed scanning and a finer scanning resolution, but also provide the higher accuracy required for the real time measurement of contaminant transport phenomena.

Appendix

Computer Program

(Coded by Visual Basic version 4 and 5)

Computer Program 1

REV Data Generator

(Coded by Visual Basic version 5)

GENREV.VBP

Project Settings

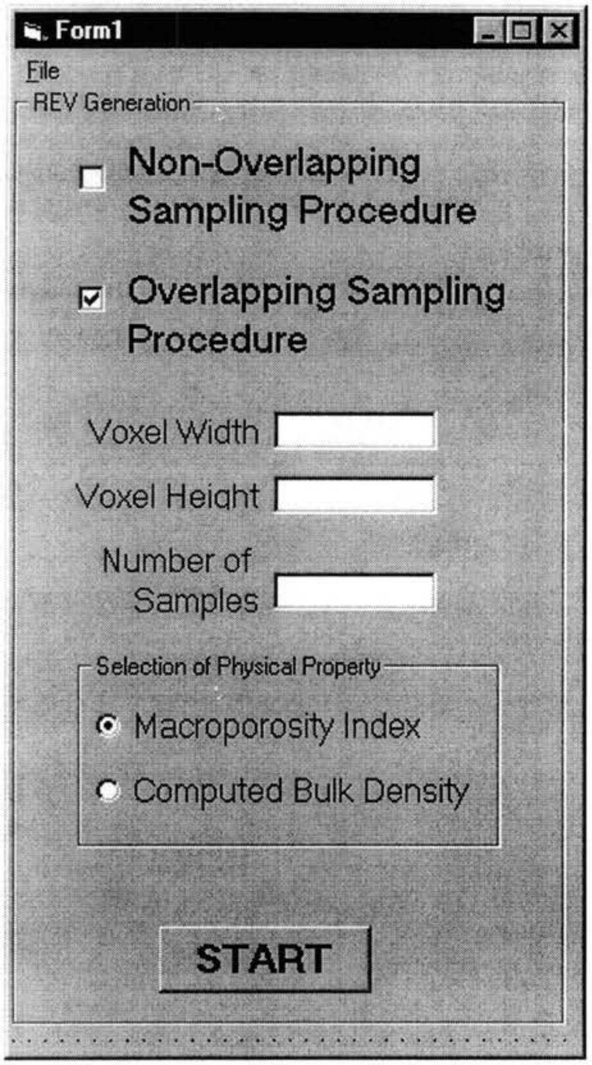
Type	Exe
IconForm	FrmREV
Startup	FrmREV
ExeName32	GenREV.exe
Command32	
Name	Project1
HelpContextID	0
CompatibleMode	0
MajorVer	1
MinorVer	0
RevisionVer	0
AutoIncrementVer	0
ServerSupportFiles	0
VersionCompanyName	Oklahoma State University
CompilationType	0
OptimizationType	0
FavorPentiumPro(tm)	0
CodeViewDebugInfo	0
NoAliasing	0
BoundsCheck	0
OverflowCheck	0
FlPointCheck	0
FDIVCheck	0
UnroundedFP	0
StartMode	0
Unattended	0
ThreadPerObject	0
MaxNumberOfThreads	1

Project References

Reference	OLE Automation

FRMREV.FRM

Mod Date	Thu Oct 16 17:55:21 1997
Size	8266



Declarations

```
Attribute VB_Name = "FrmREV"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
```

Menu

Caption	Shortcut	Name

&File	mnuFile
Set&Path and Load Data	mnuSetAndLoad

Subroutines

Command1_Click

Qualifiers: Private

```
Private Sub Command1_Click()
'The size of whole image is 17.5 cm, consisting of 14,400 pixels (120x120).
'The max. diameter of sample core is 14 cm.
'The largest length of the squared core sample is 10 cm, with area of 100 cm2.
'The width per pixel is 0.15 cm (17.5 cm/120 pixel), whereas the height is 0.3 cm.

' Max diameter of the core is 14 cm

VoxelWidth = Val(FrmREV.TxtVoxelWidth)
VoxelHeight = Val(FrmREV.TxtVoxelHeight)
NumSample = Val(FrmREV.TxtNumSample)
Call Load3DFile

    If FrmREV.ChkNonOver Then
        Call NonOverlap
    ElseIf FrmREV.ChkOver Then
        Call Overlap
    End If


End Sub
```

mnuSetAndLoad_Click

Qualifiers: Private

```
Private Sub mnuSetAndLoad_Click()
    frmSetDataPath.Show vbModal
    Refresh

    'MsgBox ("set path is OK")      ok
End Sub
```

Text1_Change

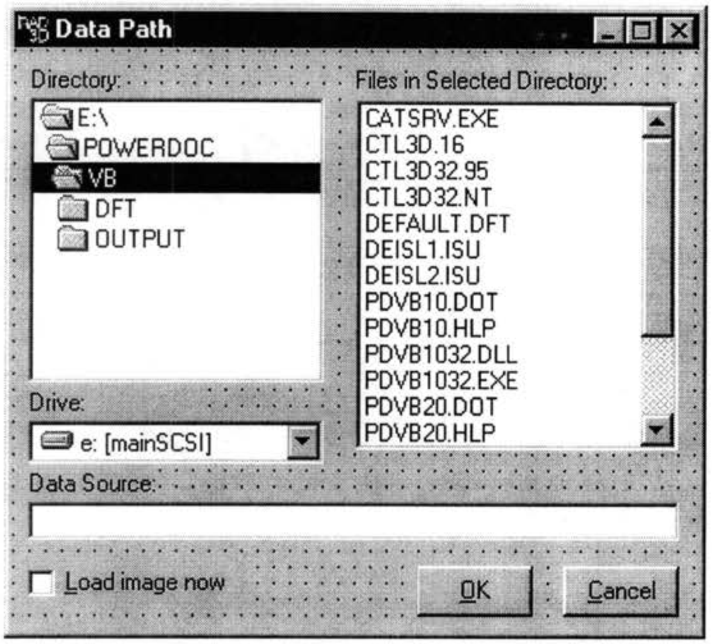
Qualifiers: Private

```
Private Sub Text1_Change()

End Sub
```

SETDPATH.FRM

Mod Date Thu Aug 28 11:21:04 1997
Size 3759



Declarations

```
Attribute VB_Name = "frmSetDataPath"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
```

Subroutines

cmdCancel_Click

Qualifiers: Private

```
Private Sub cmdCancel_Click()

    Hide

End Sub
```

cmdOK_Click

Qualifiers: Private

```
Private Sub cmdOK_Click()

    DataPath = Dir1
    Hide

End Sub
```

Dir1_Change

Qualifiers: Private

```
Private Sub Dir1_Change()

    txtDataPath = Dir1
    File1 = Dir1

End Sub
```

Drive1_Change

Qualifiers: Private

```
Private Sub Drive1_Change()

    Dir1 = Drive1
    txtDataPath = Dir1

End Sub
```

Form_Load

Qualifiers: Private

```
Private Sub Form_Load()

    If DataPath = "" Then
        DataPath = App.Path
    End If

    Drive1 = DataPath
    Dir1 = DataPath
    File1.Path = DataPath
    txtDataPath = DataPath

End Sub
```

txtDataPath_Change

Qualifiers: Private

```
Private Sub txtDataPath_Change()

    Dir1 = txtDataPath
    Drive1 = txtDataPath
    File1 = txtDataPath

End Sub
```

GENREV.BAS

Mod Date Thu Oct 16 23:13:23 1997
Size 13461

Declarations

```
Attribute VB_Name = "GenRev"
Option Explicit
Global Const PI = 3.14159
Global Const NumOfCorner = 8 ' number of volume sampled
Global VoxelHeight          ' cm   , diameter of gamma-ray beam
Global VoxelWidth           ' cm   , diameter of gamma-ray beam 17.5(cm)/120(pixel)
Global NumSample            ' number of samples required per volume
Global nMax As Integer, nMin As Integer
Global radData() As Double
Global NumFiles As Integer
Global Size As Long
Global DataPath As String
Global FilePath As String
```

Subroutines

Load3DFile

Qualifiers: Public

```
Sub Load3DFile()
    Dim file As String
    Dim pos As Integer
    Dim N As Integer
    Dim FileNum
    Dim NameStr As String, str As String

    FilePath = IIf(Right$(DataPath, 1) <> "\", DataPath & "\*.rad", DataPath &
"*.rad")

    file = LCase(Dir$(FilePath))
    'first, try to get the number of files
    pos = InStr(file, ".rad")
    NameStr = Left$(file, pos - 3)
    nMax = 0
    nMin = 999
    NumFiles = 0
    Do While file <> ""
        NumFiles = NumFiles + 1
        pos = InStr(file, ".rad")
        N = Val(Mid$(file, pos - 2, 2))
        If N > nMax Then nMax = N
        If N < nMin Then nMin = N
        file = Dir$() 'get next file in directory
    Loop
    file = NameStr & Format$(nMin, "00") & ".rad"
    FilePath = IIf(Right$(DataPath, 1) <> "\", DataPath & "\" & file, DataPath &
file)
    FileNum = FreeFile
    Open FilePath For Input As FileNum
    Line Input #FileNum, str
    Size = Val(str)
    Close #FileNum
```

```
    ReDim radData(NumFiles, Size, Size) As Double

    Dim i As Long, X As Long, Y As Long
    Dim tmp As Long, rad As Double
    Dim radMax As Integer
    Dim radMin As Integer

    radMax = -9999#
    radMin = 9999#

    N = 0
    For i = nMin To nMax
        file = NameStr & Format$(i, "00") & ".rad"
        FilePath = IIf(Right$(DataPath, 1) <> "\", DataPath & "\" & file, DataPath &
file)
        file = Dir$(FilePath)
        If file <> "" Then
            N = N + 1
            FileNum = FreeFile
            Open FilePath For Input As FileNum
            Line Input #FileNum, str
            tmp = Val(str)
            If tmp = Size Then
                For Y = 1 To Size
                    For X = 1 To Size
                        Line Input #FileNum, str
                        rad = Val(str)
                        radData(N, Y, X) = rad
                    Next X
                Next Y
            Else
              Close #FileNum
            End If
        End If
    Next i
    'Dim tempRadData() As Double
    'redimtempRadData(NumFiles, Size, Size) As Double
    'tempRadData
    'For i = nMin To nMax
    '    For Y = 1 To Size
    '        For X = 1 To Size
    '            radData(i, Y, X) = rad
    '        Next X
    '    Next Y
    'Next i

End Sub
```

Overlap

Qualifiers: Public

```
Sub Overlap()

    Dim Outfile
    Dim FileNum
    Dim i, j, k, ii, jj, kk As Integer
    Dim TempX, TempY, TempLayer As Integer
    Dim Height, DataSize As Long
    Dim ExpandFactor As Integer
    Dim Area As Double

    Dim Volume() As Double
    ReDim Volume(1 To NumSample) As Double
    Dim DataNum() As Long
    ReDim DataNum(1 To NumSample) As Long

    Dim DFactor() As Integer
    Dim XFactor() As Integer
    Dim YFactor() As Integer
    Dim StartX() As Integer
```

```
    Dim StartY() As Integer
    Dim EndX() As Integer
    Dim EndY() As Integer
    Dim StartLayer() As Integer
    Dim EndLayer() As Integer

    ReDim DFactor(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim XFactor(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim YFactor(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim StartX(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim StartY(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim EndX(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim EndY(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim StartLayer(1 To NumOfCorner, 1 To NumSample) As Integer
    ReDim EndLayer(1 To NumOfCorner, 1 To NumSample) As Integer

    Dim dfT() As Double
    Dim dfTr() As Double
    Dim dfE() As Double
    Dim SST()  As Double
    Dim SSTr() As Double
    Dim SSE() As Double
    Dim MSTr() As Double
    Dim MSE() As Double
    Dim FValue() As Double
    Dim TrMean() As Double
    Dim TotalMean() As Double

    ReDim dfT(1 To NumSample) As Double
    ReDim dfTr(1 To NumSample) As Double
    ReDim dfE(1 To NumSample) As Double
    ReDim SST(1 To NumSample) As Double
    ReDim SSTr(1 To NumSample) As Double
    ReDim SSE(1 To NumSample) As Double
    ReDim MSTr(1 To NumSample) As Double
    ReDim MSE(1 To NumSample) As Double
    ReDim FValue(1 To NumSample) As Double
    ReDim TrMean(1 To NumOfCorner, 1 To NumSample) As Double
    ReDim TotalMean(1 To NumSample) As Double

    Dim TempMean As Double
    Dim tempRadData As Double

    Dim MaxDataNum As Long
    Dim MaxVolume As Double
    Dim MaxTrMean As Double

    Dim tempT, tempTr As Double

'open output file ----------------------------------------
    Outfile = DataPath & "\output.dat"
    FileNum = FreeFile
    Open Outfile For Output As FileNum
'------------------------------------------------------------
    Dim MaxLayer As Integer

    MaxLayer = nMax - nMin + 1

    'define solid part to zero if applying macroporosity index
    If FrmREV.OptMI Then
        For ii = 1 To MaxLayer
            For jj = 1 To Size
                For kk = 1 To Size
                    If (radData(ii, jj, kk) <> 1#) Then
                        radData(ii, jj, kk) = 0#
                    End If
                Next kk
            Next jj
        Next ii
    Else
        For ii = 1 To MaxLayer
            For jj = 1 To Size
                For kk = 1 To Size
                    radData(ii, jj, kk) = radData(ii, jj, kk) * 135.7
                Next kk
            Next jj
```

```
        Next ii
    End If

    ExpandFactor = Int(Size / NumSample)
    For j = 1 To NumSample
        For i = 1 To NumOfCorner
        DataSize = j * ExpandFactor
        Height = j
        Area = DataSize * DataSize * VoxelWidth * VoxelWidth
        Volume(j) = Area * Height * VoxelHeight
        DataNum(j) = DataSize * DataSize * Height

   Select Case (i)
    Case 1:
        StartLayer(i, j) = 1
        DFactor(i, j) = 1
        StartX(i, j) = Size
        StartY(i, j) = 1
        EndX(i, j) = Size - DataSize + 1
        EndY(i, j) = 1 + DataSize - 1
    Case 2:
        StartLayer(i, j) = 1
        DFactor(i, j) = 1
        StartX(i, j) = 1
        StartY(i, j) = 1
        EndX(i, j) = 1 + DataSize - 1
        EndY(i, j) = 1 + DataSize - 1
    Case 3:
        StartLayer(i, j) = 1
        DFactor(i, j) = 1
        StartX(i, j) = 1
        StartY(i, j) = Size
        EndX(i, j) = 1 + DataSize - 1
        EndY(i, j) = Size - DataSize + 1
    Case 4:
        StartLayer(i, j) = 1
        DFactor(i, j) = 1
        StartX(i, j) = Size
        StartY(i, j) = Size
        EndX(i, j) = Size - DataSize + 1
        EndY(i, j) = Size - DataSize + 1
    Case 5:
        StartLayer(i, j) = NumFiles
        DFactor(i, j) = -1
        StartX(i, j) = 1
        StartY(i, j) = 1
        EndX(i, j) = 1 + DataSize - 1
        EndY(i, j) = 1 + DataSize - 1
    Case 6:
        StartLayer(i, j) = NumFiles
        DFactor(i, j) = -1
        StartX(i, j) = Size - DataSize + 1
        StartY(i, j) = 1
        EndX(i, j) = Size
        EndY(i, j) = 1 + DataSize - 1
    Case 7:
        StartLayer(i, j) = NumFiles
        DFactor(i, j) = -1
        StartX(i, j) = 1
        StartY(i, j) = Size - DataSize + 1
        EndX(i, j) = 1 + DataSize - 1
        EndY(i, j) = Size
    Case 8:
         StartLayer(i, j) = NumFiles
         DFactor(i, j) = -1
         StartX(i, j) = Size - DataSize + 1
         StartY(i, j) = Size - DataSize + 1
         EndX(i, j) = Size
         EndY(i, j) = Size
    End Select
    EndLayer(i, j) = StartLayer(i, j) + (j - 1) * DFactor(i, j)

    'Adjust the begin and end points that "End > Start" is always the case
    If (EndX(i, j) < StartX(i, j)) Then
        TempX = StartX(i, j)
        StartX(i, j) = EndX(i, j)
```

```
        EndX(i, j) = TempX
    End If
    If (EndY(i, j) < StartY(i, j)) Then
        TempY = StartY(i, j)
        StartY(i, j) = EndY(i, j)
        EndY(i, j) = TempY
    End If
    If (EndLayer(i, j) < StartLayer(i, j)) Then
        TempLayer = StartLayer(i, j)
        StartLayer(i, j) = EndLayer(i, j)
        EndLayer(i, j) = TempLayer
    End If
    Next i
Next j

 'test data set 10-16-97
 'NumOfCorner = 2
 'NumSample = 1
 '   StartLayer(1, 1) = 1
 '   StartX(1, 1) = 1
 '   StartY(1, 1) = 1
 '   EndLayer(1, 1) = 1
 '   EndX(1, 1) = 1
 '   EndY(1, 1) = 6
 '   StartLayer(2, 1) = 2
 '   StartX(2, 1) = 1
 '   StartY(2, 1) = 1
 '   EndLayer(2, 1) = 2
 '   EndX(2, 1) = 1
 '   EndY(2, 1) = 6
 '   radData(1, 1, 1) = 6.1
 '   radData(1, 1, 2) = 7.1
 '   radData(1, 1, 3) = 7.8
 '   radData(1, 1, 4) = 6.9
 '   radData(1, 1, 5) = 7.6
 '   radData(1, 1, 6) = 8.2
 '   radData(2, 1, 1) = 9.1
 '   radData(2, 1, 2) = 8.2
 '   radData(2, 1, 3) = 8.6
 '   radData(2, 1, 4) = 6.9
 '   radData(2, 1, 5) = 7.5
 '   radData(2, 1, 6) = 7.9
 '   DataNum(1) = 6


 'read in 8 3-D block having the same volume
 'eg. vol(1)-corner 1 ~ 8, vol(2)-corner 1 ~ 8, etc.

    For j = 1 To NumSample
        For i = 1 To NumOfCorner
            tempRadData = 0#
            For ii = StartLayer(i, j) To EndLayer(i, j)
                For jj = StartX(i, j) To EndX(i, j)
                    For kk = StartY(i, j) To EndY(i, j)
                            tempRadData = tempRadData + radData(ii, jj, kk)
                    Next kk
                Next jj
            Next ii
    ' ..............................................................._
        'calculate the treatment mean -  X.
            TrMean(i, j) = CDbl(tempRadData) / CDbl(DataNum(j))
            Print #FileNum, Format(i, "##"), Format(DataNum(j), "#####"),
Format(Volume(j), "###0.000"), Format(TrMean(i, j), "##0.00000")
        Next i
    Next j

    'define the max volume size and its property - we collect 8 points for double
checking.
    For i = 1 To NumOfCorner
        tempRadData = 0#
            For ii = 1 To MaxLayer
                For jj = 1 To Size
                    For kk = 1 To Size
                        tempRadData = tempRadData + radData(ii, jj, kk)
                    Next kk
                Next jj
```

```
            Next ii
        MaxDataNum = MaxLayer * Size * Size
        MaxVolume = MaxDataNum * VoxelWidth * VoxelWidth * VoxelHeight
        MaxTrMean = tempRadData / MaxDataNum
        Print #FileNum, Format(i, "##"), Format(MaxDataNum, "#####"),
Format(MaxVolume, "###0.000"), Format(MaxTrMean, "##0.00000")
    Next i

' ....................................................
    'calculate the total mean -  X..= (sum of TrMean) / NumOfCorner
    'NumSample = 31
    For j = 1 To NumSample
        For i = 1 To NumOfCorner
            TempMean = TempMean + TrMean(i, j)
        Next i
        TotalMean(j) = TempMean / NumOfCorner
        TempMean = 0#
    Next j


' ....................................................
    'calculate the sum of squares of treatment -  SSTr
    'calculate the sum of squares of error -  SSE
    'calculate the sum of squares of total -  SST
    '-------------------------------------------------
    'Detailed formulation see Devore (1995), p. 398
    '-------------------------------------------------
    tempT = 0#
    tempTr = 0#
    For j = 1 To NumSample
        For i = 1 To NumOfCorner
            For ii = StartLayer(i, j) To EndLayer(i, j)
                For jj = StartX(i, j) To EndX(i, j)
                    For kk = StartY(i, j) To EndY(i, j)
                           tempT = tempT + radData(ii, jj, kk) * radData(ii, jj, kk)
                    Next kk
                Next jj
            Next ii
               tempTr = tempTr + (TrMean(i, j) - TotalMean(j)) * (TrMean(i, j) -
TotalMean(j))
        Next i
        SST(j) = tempT - TotalMean(j) * TotalMean(j) * (CDbl(NumOfCorner) *
CDbl(DataNum(j)))
        SSTr(j) = tempTr * CDbl(DataNum(j))
        SSE(j) = SST(j) - SSTr(j)
        tempTr = 0#
        tempT = 0#
    Next j

    'degree of freedom
    'Mean Square of Treatment
    'Mean Square of Total
    'F-value
    For j = 1 To NumSample
        dfTr(j) = CDbl(NumOfCorner - 1)
        dfE(j) = CDbl(NumOfCorner) * CDbl((DataNum(j) - 1))
        dfT(j) = dfE(j) + dfTr(j)
        MSTr(j) = SSTr(j) / dfTr(j)
        MSE(j) = SSE(j) / dfE(j)
        FValue(j) = MSTr(j) / MSE(j)

        Print #FileNum,
        Print #FileNum, Format(dfTr(j), "0000"), Format(SSTr(j), "###000.0000"),
Format(MSTr(j), "###00.0000"), Format(FValue(j), "###00.000")
        Print #FileNum, Format(dfE(j), "0000"), Format(SSE(j), "###000.0000"),
Format(MSE(j), "###00.0000")
        Print #FileNum, Format(dfT(j), "0000"), Format(SST(j), "###000.0000"),
        Print #FileNum,
        Print #FileNum,
    Next j

Close #FileNum    ' Close file.
MsgBox ("completed")
'Unload FrmREV

End Sub
```

Procedure List

Procedure	Module	Returns	Arg	Type
cmdCancel_Click	SETDPATH.FRM		(None)	(N/A)
cmdCancel_Click	SETDPATH.FRM		(None)	(N/A)
cmdOK_Click	SETDPATH.FRM		(None)	(N/A)
Command1_Click	FRMREV.FRM		(None)	(N/A)
Dir1_Change	SETDPATH.FRM		(None)	(N/A)
Drive1_Change	SETDPATH.FRM		(None)	(N/A)
Form_Load	SETDPATH.FRM		(None)	(N/A)
Load3Dfile	GENREV.BAS		(None)	(N/A)
mnuSetAndLoad_Cl ick	FRMREV.FRM		(None)	(N/A)
NonOverlap	GENREV.BAS		(None)	(N/A)
Overlap	GENREV.BAS		(None)	(N/A)
Text1_Change	FRMREV.FRM		(None)	(N/A)
txtDataPath_Change	SETDPATH.FRM		(None)	(N/A)

Procedure Calling List

Procedure	Module	Calls	Module
Command1_Click	FRMREV.FRM	Load3DFile	GENREV.BAS
		NonOverlap	GENREV.BAS
		Overlap	GENREV.BAS

Procedure Called By List

Procedure	Module	Called By	Module
Load3DFile	GENREV.BAS	Command1_Click	FRMREV.FRM
NonOverlap	GENREV.BAS	Command1_Click	FRMREV.FRM
Overlap	GENREV.BAS	Command1_Click	FRMREV.FRM

Index

Computer Program 2

SST Algorithm

(Coded by Visual Basic version 4)

Table of contents

SCDP3.VBP

Project Settings

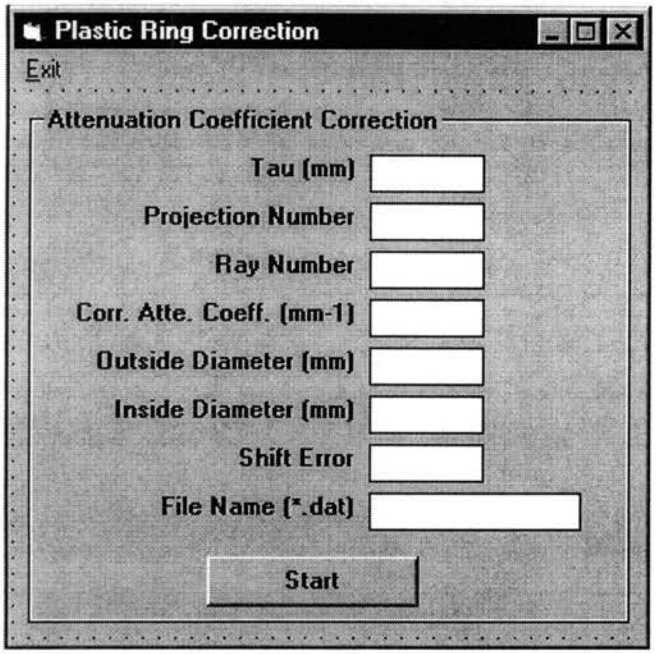
ProjWinSize	83,571,223,293
ProjWinShow	2
IconForm	frmMinMax
HelpFile	
Title	RADON
ExeName32	SCDP2 15.exe
ExeName	SCDP1 1.EXE
Name	SCDP2
HelpContextID	0
StartMode	0
VersionCompatible32	0
MajorVer	1
MinorVer	0
RevisionVer	0
AutoIncrementVer	0
ServerSupportFiles	0
VersionCompanyName	Oklahoma State University

Project References

Object	GRID32.OCX
Object	ANIBTN32.OCX
Object	COMDLG32.OCX
Object	GAUGE32.OCX
Object	GRAPH32.OCX
Object	KEYSTA32.OCX
Object	mscomm32.ocx
Object	msmask32.ocx
Object	MSOUTL32.OCX
Object	picclp32.ocx
Object	SPIN32.OCX
Object	THREED32.OCX
Reference	Microsoft DAO 2.5/3.0 Compatibility Library
Object	sndrec32.exe
Object	avi

ATTCOEFF.FRM

Mod Date	Fri Oct 04 12:45:17 1996
Size	9309



Declarations

```
Attribute VB_Name = "frmCorrect"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
```

Menu

Caption	Shortcut	Name
&Exit		mnuExit

Subroutines

Command1_Click

Qualifiers: Private

```
Private Sub Command1_Click()

    Tau = Val(txtTau.Text)
    ProjNum = Val(txtProjNum.Text)
    RayNum = Val(txtRayNum.Text)
    AttenCoeff = Val(txtatten.Text)
    OutDiameter = Val(txtOD.Text)
```

```
    InDiameter = Val(txtID.Text)
    ShiftError = Val(txtShiftError.Text)
    'txtInFile.Text = UCase$(frmGetFile.Text1)
    'InputFilename = txtInFile.Text
    If InputFile = "" Then
        Beep
        MsgBox "Incorrect Input file name"
        Exit Sub
    End If

    '**** Check whether the Input File has the extension .DAT ****
    StrLen = Len(InputFile)
    Debug.Print InputFile
    If UCase$(Mid$(InputFile, StrLen - 3, 4)) <> ".DAT" Then
        Beep
        MsgBox "Error: Input file name must end in .DAT"
        Exit Sub
    End If
    'ReDim Intens0(1 To ProjNum) As Long
    'Call ReadInputFile(InputFile)
    Call CorrectattenCoeff(InputFile)

End Sub
```

mnuExit_Click

Qualifiers: Private

```
Private Sub mnuExit_Click()
    frmCorrect.Hide
    frmRadon.Show
    Unload frmCorrect
End Sub
```

Txtatten_GotFocus

Qualifiers: Private

```
Private Sub Txtatten_GotFocus()
    txtatten.SelStart = 0
    txtatten.SelLength = 65000

End Sub
```

TxtID_GotFocus

Qualifiers: Private

```
Private Sub TxtID_GotFocus()
    txtID.SelStart = 0
    txtID.SelLength = 65000
End Sub
```

TxtInfile_DblClick

Qualifiers: Private

```
Private Sub TxtInfile_DblClick()

    Dim temp$

    frmGetFile.Caption = "Input File Name"
    frmGetFile.FileTypes.AddItem "Data Files (*.DAT)"
    frmGetFile.Show MODAL
    temp$ = frmGetFile.FullPath.Text
    If temp$ <> "" Then
        txtInFile.Text = UCase$(frmGetFile.Text1)
        'We set a global variable here
        InputFile = UCase$(frmGetFile.FullPath.Text)
    End If

End Sub
```

TxtInfile_GotFocus

Qualifiers: Private

```
Private Sub TxtInfile_GotFocus()
    txtInFile.SelStart = 0
    txtInFile.SelLength = 65000

End Sub
```

TxtOD_GotFocus

Qualifiers: Private

```
Private Sub TxtOD_GotFocus()
    txtOD.SelStart = 0
    txtOD.SelLength = 65000
End Sub
```

TxtProjNum_GotFocus

Qualifiers: Private

```
Private Sub TxtProjNum_GotFocus()
    txtProjNum.SelStart = 0
    txtProjNum.SelLength = 65000
End Sub
```

TxtRayNum_GotFocus

Qualifiers: Private

```
Private Sub TxtRayNum_GotFocus()
    txtRayNum.SelStart = 0
    txtRayNum.SelLength = 65000
End Sub
```

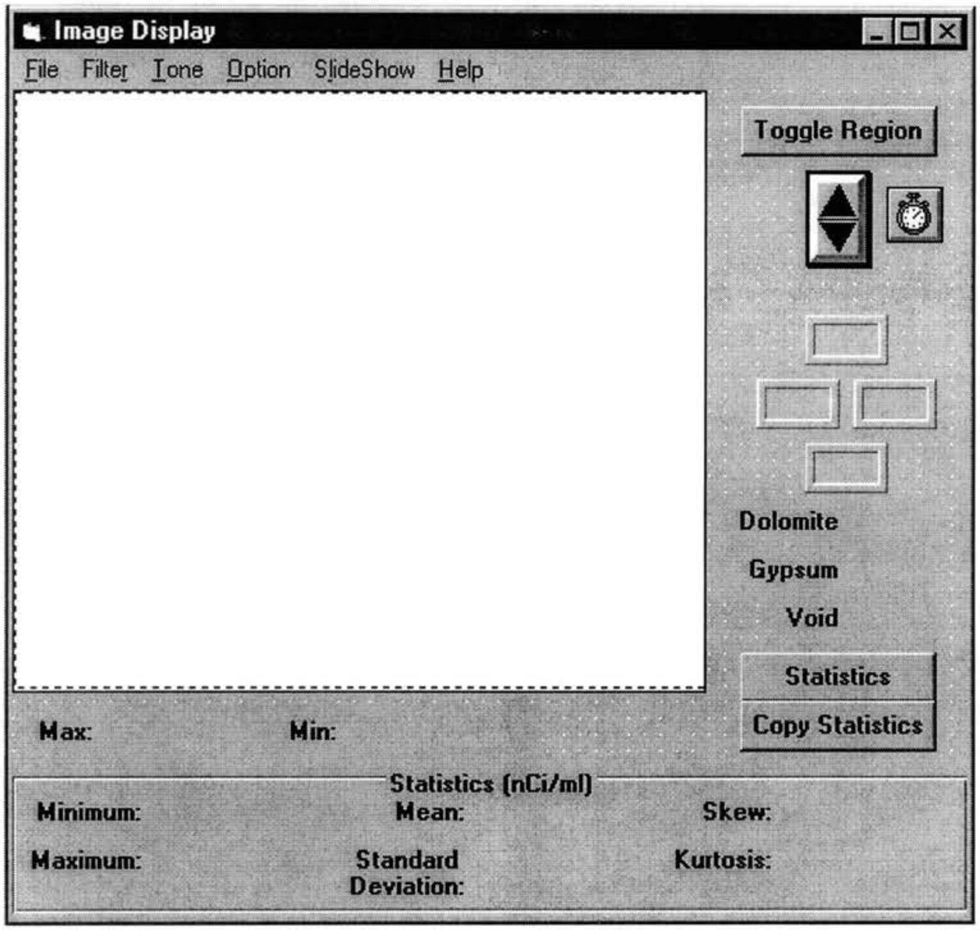
TxtTau_GotFocus

Qualifiers: Private

```
Private Sub TxtTau_GotFocus()
    txtTau.SelStart = 0
    txtTau.SelLength = 65000
End Sub
```

DISPLAY.FRM

Mod Date Tue Sep 16 10:57:49 1997
Size 59946



Declarations

```
Attribute VB_Name = "frmDisplay"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
Dim Palette() As Integer
Dim BmpArray() As Integer
Dim BmpSize As Integer
Dim BmpFileName As String
'Variables holding and describing the image Data

Dim NV As Integer
Dim Mu() As Single
Dim DisplayFormCaption As String
Dim BlackThreshold As Integer
Dim WhiteThreshold As Integer
Dim PelsPerScaleWidth
Dim PelsPerScaleHeight
Dim RegionVisible As Integer
Dim DrawRegion As Integer
```

```
Dim RedrawRegion As Integer
Dim ImageVisible As Integer
Dim SizeChangeIncrement
Dim RegionRadiusIncrement As Integer
Dim RegionXPos As Integer
Dim RegionYPos As Integer
Dim RegionRadius As Integer
Dim OldRegionXPos As Integer
Dim OldRegionYPos As Integer
```

Menu

Caption	Shortcut	Name
&File		mnuFile
&Load Image		mnuLoadImage
E&xit		mnuExit
Filte&r		mnuFilter
&High Pass Filter	^H	mnuHighPassFilter
&Low Pass Filter	^L	mnuLowPassFilter
&Tone		mnuTone
&Black Up	^B	mnuBlackUp
&White Down	^W	mnuWhiteDown
&Option		mnuOption
S&lideShow		mnuSlideShow
&Help		mnuHelp
File		mnuFileHelp
Filter		mnuFilterHelp
Tone		mnuToneHelp
Toggle Region		mnuToggleRegionHelp
Size Region		mnuSizeRegionHelp
Statistics		mnuStatisticsHelp
Copy Statistics		mnuCopyStatisticsHelp

Subroutines

CalculateMinMaxInc

Qualifiers:	Private		
Arguments:	Mu	Single	By Ref.
	NV	Integer	By Value
	MuMin	Single	By Ref.
	MuMax	Single	By Ref.
	MuInc	Single	By Ref.

```
Private Sub CalculateMinMaxInc(Mu() As Single, ByVal NV As Integer, MuMin As Single,
MuMax As Single, MuInc As Single)
    'Given an array, this routine calculates the Minimum, the Maximum and the average
    'increment from the Minimum to the Maximum over the length of the array

    'Global variables used
    'ImgMin
    'ImgMax

    Dim I As Integer
    Dim j As Integer

'Remarked by hsieh 6-29-96
'Define max and min from keystroke
```

```
    If SelfDefineIndex <> 1 Then
        MuMin = Mu(1, 1)
        MuMax = Mu(1, 1)

        For I = 1 To NV
            For j = 1 To NV
                If Mu(I, j) < MuMin Then
                    MuMin = Mu(I, j)
                End If
                If Mu(I, j) > MuMax Then
                    MuMax = Mu(I, j)
                End If
            Next j
        Next I
        MuRealMin = MuMin
        MuRealMax = MuMax
    Else
        MuMin = ImgMin
        MuMax = ImgMax
        For I = 1 To NV
            For j = 1 To NV
                If Mu(I, j) < MuMin Then
                    Mu(I, j) = MuMin
                End If
                If Mu(I, j) >= MuMax Then
                    Mu(I, j) = MuMax
                End If
            Next j
        Next I

    End If

    MuInc = (MuMax - MuMin) / 255
End Sub
```

cmdCopyStatistics_Click

Qualifiers: Private

```
Private Sub cmdCopyStatistics_Click()
    'Routine to copy the values in the Statistics box, into the clipboard

    'Global variables used
    '   None

    Const CF_TEXT = 1
    Dim ClipStr As String

    ClipStr = ""
    ClipStr = ClipStr & lblMinimumVal.Caption & Chr$(9)
    ClipStr = ClipStr & lblMaximumVal.Caption & Chr$(9)
    ClipStr = ClipStr & lblMeanVal.Caption & Chr$(9)
    ClipStr = ClipStr & lblStandardDeviationVal.Caption & Chr$(9)
    ClipStr = ClipStr & lblSkewVal.Caption & Chr$(9)
    ClipStr = ClipStr & lblKurtosisVal.Caption & Chr$(9)

    ClipStr = ClipStr & pnlRegionLeft.Caption & Chr$(9)
    ClipStr = ClipStr & pnlRegionTop.Caption & Chr$(9)
    ClipStr = ClipStr & pnlRegionRight.Caption & Chr$(9)
    ClipStr = ClipStr & pnlRegionBottom.Caption

    Clipboard.Clear
    Clipboard.SetText ClipStr, CF_TEXT

End Sub
```

cmdStatistics_Click

Qualifiers: Private

```
Private Sub cmdStatistics_Click()
    'Routine to calculate the statistics

    'Global variables used
    '   - RegionXPos
    '   - RegionYPos
    '   - RegionRadius
    '   - Mu()

    Dim Minimum As Single
    Dim Maximum As Single
    Dim Mean As Double
    Dim StandardDeviation As Double
    Dim Skew As Double
    Dim Kurtosis As Double
    Dim temp As Double

    Dim Sum As Single
    Dim DistancePower2 As Double
    Dim DistancePower3 As Double
    Dim DistancePower4 As Double
    Dim DataArray() As Single
    Dim DataArraySize As Long
    Dim MaxDataArraySize As Long

    Dim I As Integer, j As Integer
    Dim Ix As Integer, Iy As Integer
    Dim RegionLeft As Integer
    Dim RegionRight As Integer
    Dim RegionTop As Integer
    Dim RegionBottom As Integer
    Dim DistanceFromCenterSquared As Long

    Call FindRegionBoundaries(RegionXPos, RegionYPos, RegionRadius, RegionLeft,
RegionRight, RegionTop, RegionBottom)
    MaxDataArraySize = (RegionRight - RegionLeft + 1) * (RegionBottom - RegionTop +
1)
    ReDim DataArray(1 To MaxDataArraySize)
    DataArraySize = 0

    For Iy = RegionTop To RegionBottom
        For Ix = RegionLeft To RegionRight
            DistanceFromCenterSquared = (((Ix - 0.5) - RegionXPos) ^ 2) + (((Iy -
0.5) - RegionYPos) ^ 2)
            If DistanceFromCenterSquared <= (RegionRadius ^ 2) Then
                DataArraySize = DataArraySize + 1
                DataArray(DataArraySize) = Mu(Iy, Ix)
            End If
        Next Ix
    Next Iy

    'Now start calculating the statistics
    ReDim DataArraySquared(1 To DataArraySize)
    ReDim DataArrayCubed(1 To DataArraySize)
    ReDim DataArrayQuadrupled(1 To DataArraySize)

    Debug.Print "DataArraySize = "; DataArraySize

    Sum = 0
    Minimum = DataArray(1)
    Maximum = DataArray(1)
    For I = 1 To DataArraySize
        Sum = Sum + DataArray(I)
        If DataArray(I) < Minimum Then
            Minimum = DataArray(I)
        End If
        If DataArray(I) > Maximum Then
            Maximum = DataArray(I)
        End If
    Next I

    Mean = Sum / DataArraySize
```

```
    DistancePower2 = 0
    DistancePower3 = 0
    DistancePower4 = 0
    For I = 1 To DataArraySize
        DistancePower2 = DistancePower2 + ((DataArray(I) - Mean) ^ 2)
        DistancePower3 = DistancePower3 + ((DataArray(I) - Mean) ^ 3)
        DistancePower4 = DistancePower4 + ((DataArray(I) - Mean) ^ 4)
    Next I

    StandardDeviation = Sqr(DistancePower2 / (DataArraySize - 1))
    If (StandardDeviation = 0) Then
        Skew = 0
        Kurtosis = 0
    Else
        Skew = (DataArraySize * DistancePower3) / ((DataArraySize - 1) *
(DataArraySize - 2) * (StandardDeviation ^ 3))
        'We need to split up the kurtosis calculation to
        'prevent an overflow
        temp = (DistancePower4 * (DataArraySize ^ 2)) / ((DataArraySize - 1) *
(DataArraySize - 2))
        Kurtosis = temp / ((DataArraySize - 3) * (StandardDeviation ^ 4))
    End If

    lblMinimumVal.Caption = Format$(Minimum, "####0.0#######")
    lblMaximumVal.Caption = Format$(Maximum, "####0.0#######")
    lblMeanVal.Caption = Format$(Mean, "####0.0#######")
    lblStandardDeviationVal.Caption = Format$(StandardDeviation, "####0.0#######")
    lblSkewVal.Caption = Format$(Skew, "####0.0#######")
    lblKurtosisVal.Caption = Format$(Kurtosis, "####0.0#######")
End Sub
```

cmdToggleRegion_Click

Qualifiers: Private

```
Private Sub cmdToggleRegion_Click()
    'Switch the circular region of interest On or Off

    'Global variables used
    '   - RegionVisible
    '   - RegionXPos
    '   - RegionYPos
    '   - RegionRadius

    If RegionVisible = True Then
        Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)

        RegionVisible = False
        Call DisableRegionButtons
        Call SetAllPanelsToEmptyString
    Else
        RegionVisible = True
        Call EnableRegionButtons
        Call DisplayRegionAtNewPosition(RegionXPos, RegionYPos, RegionRadius)
        Call ResetValuesInAllPanels(RegionXPos, RegionYPos, RegionRadius)
    End If
End Sub
```

DisableImageButtons

Qualifiers: Private

```
Private Sub DisableImageButtons()
    'Disable all buttons and menu items pertaining to image manipulation

    'Global variables used
```

```
    '   None

    picView.Enabled = False
    imgView.Enabled = False
    cmdToggleRegion.Enabled = False

    mnuFilter.Enabled = False
    mnuTone.Enabled = False
    Call DisableRegionButtons
End Sub
```

DisableRegionButtons

Qualifiers: Private

```
Private Sub DisableRegionButtons()
    'Disable all buttons and menu items dealing with the circular region of interest

    'Global variables used
    '   None

    spnEnlargeReduce.Enabled = False
    cmdStatistics.Enabled = False
    cmdCopyStatistics.Enabled = False
    pnlRegionTop.Enabled = False
    pnlRegionBottom.Enabled = False
    pnlRegionLeft.Enabled = False
    pnlRegionRight.Enabled = False

    fraStatistics.Enabled = False
    lblMinimum.Enabled = False
    lblMaximum.Enabled = False
    lblMean.Enabled = False
    lblStandardDeviation.Enabled = False
    lblSkew.Enabled = False
    lblKurtosis.Enabled = False

    lblMinimumVal.Enabled = False
    lblMaximumVal.Enabled = False
    lblMeanVal.Enabled = False
    lblStandardDeviationVal.Enabled = False
    lblSkewVal.Enabled = False
    lblKurtosisVal.Enabled = False

    lblMinimumVal.Caption = ""
    lblMaximumVal.Caption = ""
    lblMeanVal.Caption = ""
    lblStandardDeviationVal.Caption = ""
    lblSkewVal.Caption = ""
    lblKurtosisVal.Caption = ""

End Sub
```

DisplayRegionAtNewPosition

Qualifiers:	Private		
Arguments:	RegionXPos	Variant	By Ref.
	RegionYPos	Variant	By Ref.
	RegionRadius	Variant	By Ref.

```
Private Sub DisplayRegionAtNewPosition(RegionXPos, RegionYPos, RegionRadius)
    'Routine that displays the region of interest at the new region, specified by the
    'arguments

    'Global variables used
```

```
    '   - AspectRatio
    '   - RegionVisible

    RegionVisible = True
    picView.Circle (RegionXPos, RegionYPos), RegionRadius, , , , (1 / AspectRatio)
End Sub
```

EnableImageButtons

Qualifiers: Private

```
Private Sub EnableImageButtons()
    'Enable the buttons and menu items pertaining to image manipulation

    'Global variables used
    '   None

    picView.Enabled = True
    imgView.Enabled = True
    cmdToggleRegion.Enabled = True

    mnuFilter.Enabled = True
    mnuTone.Enabled = True
End Sub
```

EnableRegionButtons

Qualifiers: Private

```
Private Sub EnableRegionButtons()
    'Enable the buttons and menu items pertaining to the circular region of interest

    'Global variables used
    '   None

    spnEnlargeReduce.Enabled = True
    cmdStatistics.Enabled = True
    cmdCopyStatistics.Enabled = True
    pnlRegionTop.Enabled = True
    pnlRegionBottom.Enabled = True
    pnlRegionLeft.Enabled = True
    pnlRegionRight.Enabled = True

    fraStatistics.Enabled = True
    lblMinimum.Enabled = True
    lblMaximum.Enabled = True
    lblMean.Enabled = True
    lblStandardDeviation.Enabled = True
    lblSkew.Enabled = True
    lblKurtosis.Enabled = True

    lblMinimumVal.Enabled = True
    lblMaximumVal.Enabled = True
    lblMeanVal.Enabled = True
    lblStandardDeviationVal.Enabled = True
    lblSkewVal.Enabled = True
    lblKurtosisVal.Enabled = True

End Sub
```

EnlargeRegion

Qualifiers: Private

```
Private Sub EnlargeRegion()
    'Enlarge the circular region of interest by a specified amount

    'Global variables used
    '   - RegionXPos
    '   - RegionYPos
    '   - RegionRadius
    '   - RegionRadiusIncrement

    Dim RegionDiameter As Integer

    Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)

    RegionRadius = RegionRadius + RegionRadiusIncrement
    RegionDiameter = RegionRadius * 2
    If (RegionDiameter >= picView.ScaleHeight) Then
        RegionXPos = picView.ScaleWidth / 2
        RegionYPos = picView.ScaleHeight / 2
        RegionRadius = picView.ScaleHeight / 2
    End If

    Call RecalculateCenterOfRegion(RegionXPos, RegionYPos, RegionRadius)
    Call DisplayRegionAtNewPosition(RegionXPos, RegionYPos, RegionRadius)
    Call ResetValuesInAllPanels(RegionXPos, RegionYPos, RegionRadius)
End Sub
```

EraseRegionAtOldPosition

Qualifiers:	Private		
Arguments:	RegionXPos	Variant	By Ref.
	RegionYPos	Variant	By Ref.
	RegionRadius	Variant	By Ref.

```
Private Sub EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
    'Routine to erase the circular region of interest from its od position
    'This is facilitated by XOR'ing the region over itself

    'Global variables used
    '   - RegionVisible
    '   - AspectRatio

    If (RegionVisible = True) Then
        picView.Circle (RegionXPos, RegionYPos), RegionRadius, , , , (1 /
AspectRatio)
        RegionVisible = False
    End If
End Sub
```

FillFilter

Qualifiers:	Private		
Arguments:	Filter	Single	By Ref.

```
Private Sub FillFilter(Filter() As Single)
    'A temporary routine to verify the Low pass and High pass filters

    'Global variables used
    '   None

    Dim I, j
    Dim Sum As Single, Avg As Single
```

```
    ReDim Filter(1 To 9, 1 To 9)

    For I = 1 To 9
        For j = 1 To 9
            Filter(I, j) = -50
        Next j
    Next I

    For I = 2 To 8
        For j = 2 To 8
            Filter(I, j) = -10
        Next j
    Next I

    For I = 3 To 7
        For j = 3 To 7
            Filter(I, j) = 10
        Next j
    Next I

    For I = 4 To 6
        For j = 4 To 6
            Filter(I, j) = 50
        Next j
    Next I

    Filter(5, 5) = 100

    Do
        Sum = 0
        For I = 1 To 9
            For j = 1 To 9
                Sum = Sum + Filter(I, j)
            Next j
        Next I

        Debug.Print Sum
        If (Sum < 0.01) And (Sum > -0.01) Then
            Exit Do
        End If

        Avg = Sum / 81

        For I = 1 To 9
            For j = 1 To 9
                Filter(I, j) = Filter(I, j) - Avg
            Next j
        Next I
    Loop

    For I = 1 To 9
        For j = 1 To 9
            Debug.Print Filter(I, j);
        Next j
        Debug.Print
    Next I

    Debug.Print "Over"
End Sub
```

FindMaxMin

Qualifiers:	Private		
Arguments:	Mu	Single	By Ref.
	RMax	Single	By Ref.
	RMin	Single	By Ref.

```
Private Sub FindMaxMin(Mu() As Single, RMax As Single, RMin As Single)
```

```
    Dim I, j As Integer

' 9-27-96 hsieh
' fine the real max and min from Mu()

    RMin = Mu(1, 1)
    RMax = Mu(1, 1)

    For I = 1 To NV
        For j = 1 To NV
            If Mu(I, j) < RMin Then
                RMin = Mu(I, j)
            ElseIf Mu(I, j) > RMax Then
                RMax = Mu(I, j)
            End If
        Next j
    Next I

End Sub
```

FindRegionBoundaries

Qualifiers:	Private		
Arguments:	RegionXPos	Variant	By Ref.
	RegionYPos	Variant	By Ref.
	RegionRadius	Variant	By Ref.
	RegionLeft	Variant	By Ref.
	RegionRight	Variant	By Ref.
	RegionTop	Variant	By Ref.
	RegionBottom	Variant	By Ref.

```
Private Sub FindRegionBoundaries(RegionXPos, RegionYPos, RegionRadius, RegionLeft,
RegionRight, RegionTop, RegionBottom)
    'Routine to find out the bounding box of the circular region of interest

    'Global variables used
    '   None

    RegionLeft = RegionXPos - (RegionRadius - 1)
    RegionRight = RegionXPos + RegionRadius
    RegionTop = RegionYPos - (RegionRadius - 1)
    RegionBottom = RegionYPos + RegionRadius

    'XXX - We try to normalise the values. i.e., we make it
    'from 1 to Max rather than from 0 To Max

    ''This part of the code commented out on JULY95
    ''If RegionLeft <= 0 Then
    ''    RegionLeft = 1
    ''End If
    ''If RegionTop <= 0 Then
    ''    RegionTop = 1
    ''End If
    ''End of commented out code

End Sub
```

FindScaledRegionParameters

Qualifiers:	Private		
Arguments:	RegionXPos	Variant	By Ref.
	RegionYPos	Variant	By Ref.
	RegionRadius	Variant	By Ref.

ScaledRegionLeft	Variant	By Ref.
ScaledRegionRight	Variant	By Ref.
ScaledRegionTop	Variant	By Ref.
ScaledRegionBottom	Variant	By Ref.
ScaledRegionXPos	Variant	By Ref.
ScaledRegionYPos	Variant	By Ref.
ScaledRegionRadius	Variant	By Ref.

```
Private Sub FindScaledRegionParameters(RegionXPos, RegionYPos, RegionRadius,
ScaledRegionLeft, ScaledRegionRight, ScaledRegionTop, ScaledRegionBottom,
ScaledRegionXPos, ScaledRegionYPos, ScaledRegionRadius)
    'This function is used to calculate the scaled values
    'of the position of the region

    'Global variables used
    '   - PelsPerScaleWidth
    '   - PelsPerScaleHeight
    '   - AspectRatio

    Dim RegionLeft, RegionRight, RegionTop, RegionBottom
    Dim RadiusAlongWidth, RadiusAlongHeight

    'Changed - JULY95, subtracted 1 from Region Radius
    RegionLeft = RegionXPos - (RegionRadius - 1)
    RegionRight = RegionXPos + RegionRadius
    'Changed - JULY95, subtracted 1 from Region Radius
    RegionTop = RegionYPos - (RegionRadius - 1)
    RegionBottom = RegionYPos + RegionRadius

    RadiusAlongWidth = RegionRadius / (PelsPerScaleWidth * AspectRatio)
    RadiusAlongHeight = RegionRadius / PelsPerScaleHeight
    'XXX - Here we return the minimum of the two values
    If RadiusAlongWidth < RadiusAlongHeight Then
        ScaledRegionRadius = RadiusAlongWidth
    Else
        ScaledRegionRadius = RadiusAlongHeight
    End If

    ScaledRegionXPos = RegionXPos / PelsPerScaleWidth
    ScaledRegionYPos = RegionYPos / PelsPerScaleHeight
    ScaledRegionLeft = RegionLeft / PelsPerScaleWidth
    ScaledRegionRight = RegionRight / PelsPerScaleWidth
    ScaledRegionTop = RegionTop / PelsPerScaleHeight
    ScaledRegionBottom = RegionBottom / PelsPerScaleHeight

    'XXX - We try to normalise the values. i.e., we make it
    'from 1 to Max rather than from 0 To Max
    ''This part of the code commented out on JULY95
    ''If ScaledRegionLeft <= 0 Then
    ''   ScaledRegionLeft = 1
    ''End If
    ''If ScaledRegionTop <= 0 Then
    ''    ScaledRegionTop = 1
    ''End If
    ''End of commented out code

End Sub
```

Form_Load

Qualifiers: Private

```
Private Sub Form_Load()
    'Routine called when the form is loaded

    'Global variables used
    '   - DisplayFormCaption
```

```
    '    - SizeChangeIncrement
    '    - RegionRadiusIncrement
    '    - RegionVisible
    '    - ImageVisible
    '    - DrawRegion

    'Center the form on the screen
    Move (Screen.Width - Width) / 2, (Screen.Height - Height) / 2

    'Initialize all necessary Global variables
    'XXX - Change these increments to user adjustable values
    DisplayFormCaption = frmDisplay.Caption
    SizeChangeIncrement = 20
    RegionRadiusIncrement = 1
    RegionVisible = False
    ImageVisible = False
    DrawRegion = False
    'RedrawRegion = False
    Call DisableImageButtons
End Sub
```

Form_Paint

Qualifiers: Private

```
Private Sub Form_Paint()
    'Routine gets called when this form gets uncovered as a result of an expose event

    'Global variables used
    '    - RedrawRegion

    RedrawRegion = True
End Sub
```

Form_Unload

Qualifiers:	Private		
Arguments:	Cancel	Integer	By Ref.

```
Private Sub Form_Unload(Cancel As Integer)
    'Called when this from is unloaded

    'Global variables used
    '    None

    frmDisplay.Hide
    frmRadon.Show
End Sub
```

GenerateBitmapArray

Qualifiers:	Private		
Arguments:	Mu	Single	By Ref.
	NV	Integer	By Value
	BmpArray	Integer	By Ref.
	BmpSize	Integer	By Ref.

```
Private Sub GenerateBitmapArray(Mu() As Single, ByVal NV As Integer, BmpArray() As
Integer, BmpSize As Integer)
    'Generate an array that is a scaled version of another array,
```

```
    'i.e., the new array will contain values in a certain range,
    'normally this range is 0 to 255, corresponding to the range
    'of colors in the 256 color mode. The original array is Mu,
    'which is the Density array generated by the image reconstruction
    'algorithm. The Bitmap array is a scaled version of the Density
    'array (Scaled for purposes of displaying the image)

    'Global variables used
    '   None

    Dim I As Integer
    Dim j As Integer
    Dim MuMin As Single
    Dim MuMax As Single
    Dim MuInc As Single

    Call CalculateMinMaxInc(Mu(), NV, MuMin, MuMax, MuInc)

    'Now generate the Bitmap array
    BmpSize = NV
    ReDim BmpArray(1 To BmpSize, 1 To BmpSize)
    For I = 1 To BmpSize
        For j = 1 To BmpSize
            BmpArray(I, j) = Int((Mu(I, j) - MuMin) / MuInc)
        Next j
    Next I

End Sub
```

imgView_MouseDown

Qualifiers:	Private		
Arguments:	Button	Integer	By Ref.
	Shift	Integer	By Ref.
	X	Single	By Ref.
	Y	Single	By Ref.

```
Private Sub imgView_MouseDown(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    'Routine called when the user clicks the Mouse in the image display panel

    'Global variables used
    '   - RegionXPos
    '   - RegionYPos
    '   - RegionRadius
    '   - DrawRegion
    '   - RegionVisible
    '   - PelsPerScaleWidth
    '   - PelsPerScaleHeight

    If Button = LEFT_BUTTON Then
        DrawRegion = True
        Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
        'If the region was not present previously we need
        'to enable all the region buttons. Actually this
        'condition check is really not necessary, since
        'it dosen't matter if the region buttons are enabled
        'once more if they are already enabled
        If RegionVisible = False Then
            Call EnableRegionButtons
        End If
        RegionXPos = X / PelsPerScaleWidth
        RegionYPos = Y / PelsPerScaleHeight
        Call RecalculateCenterOfRegion(RegionXPos, RegionYPos, RegionRadius)
        Call DisplayRegionAtNewPosition(RegionXPos, RegionYPos, RegionRadius)
        Call ResetValuesInAllPanels(RegionXPos, RegionYPos, RegionRadius)
    End If
End Sub
```

imgView_MouseMove

Qualifiers:	Private		
Arguments:	Button	Integer	By Ref.
	Shift	Integer	By Ref.
	X	Single	By Ref.
	Y	Single	By Ref.

```
Private Sub imgView_MouseMove(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    'Called when the Mouse moves over the image display

    'Global variables used
    '    - DrawRegion
    '    - RegionVisible
    '    - RegionXPos
    '    - RegionYPos
    '    - RegionRadius
    '    - PelsPerScaleWidth
    '    - PelsPerScaleHeight

    If DrawRegion = True Then
        If RegionVisible = False Then
            Exit Sub
        End If

        Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
        RegionXPos = X / PelsPerScaleWidth
        RegionYPos = Y / PelsPerScaleHeight
        Call RecalculateCenterOfRegion(RegionXPos, RegionYPos, RegionRadius)
        Call DisplayRegionAtNewPosition(RegionXPos, RegionYPos, RegionRadius)
        Call ResetValuesInAllPanels(RegionXPos, RegionYPos, RegionRadius)
    End If
End Sub
```

imgView_MouseUp

Qualifiers:	Private		
Arguments:	Button	Integer	By Ref.
	Shift	Integer	By Ref.
	X	Single	By Ref.
	Y	Single	By Ref.

```
Private Sub imgView_MouseUp(Button As Integer, Shift As Integer, X As Single, Y As
Single)
    'Called when the mouse button is released

    'Global variables used
    '    - DrawRegion

    DrawRegion = False
End Sub
```

mnuBlackUp_Click

Qualifiers: Private

```
Private Sub mnuBlackUp_Click()
```

```
    'Routine to do a black up on the image

    'Global variables used
    '    - Mu()
    '    - NV
    '    - Palette()
    '    - BmpFileName
    '    - BlackThreshold
    '    - RegionXPos
    '    - RegionYPos
    '    - RegionRadius

    Dim I As Integer, j As Integer
    Dim temp As Single
    Dim MuMin As Single
    Dim MuMax As Single
    Dim MuInc As Single

    If BlackThreshold > 255 Then
        Exit Sub
    End If

    Debug.Print "BlackThreshold = "; BlackThreshold
    For I = 1 To 3
        For j = 0 To BlackThreshold
            Palette(I, j) = 0
        Next j
    Next I

    'Calculate MuMin, MuMax and MuInc
    Call CalculateMinMaxInc(Mu(), NV, MuMin, MuMax, MuInc)
    For I = 1 To NV
        For j = 1 To NV
            temp = Mu(I, j) - MuMin
            If temp < (MuInc * BlackThreshold) Then
                Mu(I, j) = 0
            End If
        Next j
    Next I

    WritePalette (BmpFileName)
    Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
    imgView.Picture = LoadPicture(BmpFileName)
    BlackThreshold = BlackThreshold + 10

End Sub
```

mnuCopyStatisticsHelp_Click

Qualifiers: Private

```
Private Sub mnuCopyStatisticsHelp_Click()
    'FileMsg "DISPLAY.MSG", 7
End Sub
```

mnuExit_Click

Qualifiers: Private

```
Private Sub mnuExit_Click()
    'Called when the user selects the Exit menu item

    'Global variables used
    '    - None

    frmDisplay.Hide
```

```
    frmRadon.Show
    Unload frmDisplay

End Sub
```

mnuFileHelp_Click

Qualifiers: Private

```
Private Sub mnuFileHelp_Click()
    'FileMsg "DISPLAY.MSG", 1
End Sub
```

mnuFilterHelp_Click

Qualifiers: Private

```
Private Sub mnuFilterHelp_Click()
    'FileMsg "DISPLAY.MSG", 2
End Sub
```

mnuHighPassFilter_Click

Qualifiers: Private

```
Private Sub mnuHighPassFilter_Click()
    'Routine to do a high pass filter on the image

    'Global variables used
    '    - NV
    '    - Mu()
    '    - BmpFileName
    '    - BmpSize
    '    - BmpArray()

    Dim I As Integer, j As Integer
    Dim k As Integer, L As Integer
    Dim M As Integer, N As Integer
    Dim SumZ As Single
    Dim NewTop As Integer, NewBottom As Integer
    Dim NewLeft As Integer, NewRight As Integer
    Dim NewMu() As Single
    Dim Filter() As Single
    Dim FilterSize As Integer
    Dim C10 As Single, C20 As Single
    Dim HalfFilterSize As Integer

    ReDim NewMu(1 To NV, 1 To NV)
    FilterSize = 9
    HalfFilterSize = Int(FilterSize / 2)
    C10 = 127
    C20 = 160
    ReDim Filter(1 To FilterSize, 1 To FilterSize)
    For I = 1 To FilterSize
        For j = 1 To FilterSize
            Filter(I, j) = -1
        Next j
    Next I
    Filter(HalfFilterSize + 1, HalfFilterSize + 1) = 80
```

```
    NewTop = 1 + HalfFilterSize
    NewBottom = NV - HalfFilterSize
    NewLeft = 1 + HalfFilterSize
    NewRight = NV - HalfFilterSize

    'Now construct the new filtered array
    For I = NewLeft To NewRight
        Debug.Print I
        For j = NewTop To NewBottom
            SumZ = 0
            For k = (I - HalfFilterSize) To (I + HalfFilterSize)
                For L = (j - HalfFilterSize) To (j + HalfFilterSize)
                    'We should vary M and N from 1 To FilterSize on any
                    'variation of K and L
                    'Some algebraic manipulation will ensure that
                    M = k + HalfFilterSize - I + 1
                    N = L + HalfFilterSize - j + 1
                    SumZ = SumZ + Mu(k, L) * Filter(M, N)
                Next L
            Next k
            NewMu(I, j) = C10 + (SumZ / C20)
        Next j
    Next I

    Call GenerateBitmapArray(NewMu(), NV, BmpArray(), BmpSize)
    'Now create the DIB file which will be read in
    Call WritePaletteAndBitmap(BmpFileName, BmpArray(), BmpSize)
    'Now copy NewMu into Mu
    For I = 1 To NV
        For j = 1 To NV
            Mu(I, j) = NewMu(I, j)
        Next j
    Next I
    Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
    imgView.Picture = LoadPicture(BmpFileName)

End Sub
```

mnuLoadImage_Click

Qualifiers: Private

```
Private Sub mnuLoadImage_Click()
    'Load the user specified image onto the display

    'Global variables used
    '   - RadonFileName
    '   - BmpFileName
    '   - Mu()
    '   - NV
    '   - BmpArray()
    '   - BmpSize
    '   - RegionXPos
    '   - RegionYPos
    '   - RegionRadius
    '   - ImageVisible
    '   - BlackThreshold
    '   - WhiteThreshold
    '   - PelsPerScaleWidth
    '   - PelsPerScaleHeight
    '   - DisplayFormCaption

'    On Error GoTo Errhandler
    Dim RadonFileName
    Dim FileNameExtension As String
    Dim StrLen
    Dim I As Integer, j As Integer
    Dim Number As Integer, NumberString As String * 8
    Dim FileNum, ReadPos
    Dim PaletteFileName As String
    Dim RealMaxMu      As Single
```

```
    Dim RealMinMu       As Single

    frmGetFile.Caption = "Input Radon File Name"
    frmGetFile.FileTypes.AddItem "Radon Files (*.RAD)"
    frmGetFile.FileTypes.AddItem "Error Files (*.ERR)"
    frmGetFile.FileTypes.AddItem "Spatial Files (*.SPA)"
    frmGetFile.FileTypes.AddItem "Direct Threshold Files (*.DIR)"
    frmGetFile.FileTypes.AddItem "Edge Files (*.EDG)"
    frmGetFile.FileTypes.AddItem "All Files (*.*)"
    frmGetFile.Show MODAL
    RadonFileName = frmGetFile.FullPath.Text

    'We check if the user has selected a bitmap file
    If RadonFileName = "" Then
        Exit Sub
    End If

    '**** Check whether the Input File has the extension .RAD ****
    StrLen = Len(RadonFileName)
    FileNameExtension = UCase$(Mid$(RadonFileName, StrLen - 3, 4))
    Debug.Print FileNameExtension
    'If (FileNameExtension <> ".RAD") And (FileNameExtension <> ".ERR") And
(FileNameExtension <> ".SPA") And (FileNameExtension <> ".DIR") And
(FileNameExtension <> ".EDG") Then
    '    Beep
    '    MsgBox "Error: Input file name must end in .RAD or .ERR or .SPA or .EDG or
.DIR"
    '    Exit Sub
    'End If

    'Now we construct the name of the Bmp file
    BmpFileName = Mid$(RadonFileName, 1, StrLen - 4) & ".BMP"
    'Now we delete the BMP file if it exists
    Kill BmpFileName
    'Read in the data for the Image

    Call ReadRadonFile(RadonFileName, Mu(), NV)
'9-27-96 hsieh --------------
    Call FindMaxMin(Mu(), RealMaxMu, RealMinMu)
    frmDisplay.lblRealMax.Caption = RealMaxMu
    frmDisplay.lblRealMin.Caption = RealMinMu
'-----------------------------------------------
    Debug.Print "Reading Over"
    Call GenerateBitmapArray(Mu(), NV, BmpArray(), BmpSize)

    'Now create the DIB file which will be read in
    Call WritePaletteAndBitmap(BmpFileName, BmpArray(), BmpSize)

    '11-5-96 hsieh
    Dim DoloPer, GypPer, VoidPer As Single
    Call CompPercent(Mu(), NV, DoloPer, GypPer, VoidPer)

    frmDisplay.LblDoloPercent.Caption = Format$(DoloPer, "##0.0##")
    frmDisplay.LblGypPercent.Caption = Format$(GypPer, "##0.0##")
    frmDisplay.LblVoidPercent.Caption = Format$(VoidPer, "##0.0##")

    'Next we load the Bitmap file onto the Image control
    'and also set some properties of the image and picture
    'control
    picView.ScaleWidth = NV
    picView.ScaleHeight = NV
    PelsPerScaleWidth = picView.Width / NV
    PelsPerScaleHeight = picView.Height / NV
    imgView.Left = 0
    imgView.Top = 0
    'BmpFileName = "d:\hsieh\cat\transmit\data\c3av\c3av1.bmp"
    imgView.Picture = LoadPicture(BmpFileName)
    RegionXPos = NV / 2
    RegionYPos = NV / 2
    RegionRadius = NV / 8

    ImageVisible = True
    BlackThreshold = 5
    WhiteThreshold = 250
    Call EnableImageButtons
    Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
```

```
    Call DisableRegionButtons
    frmDisplay.Caption = DisplayFormCaption & " - " & UCase$(BmpFileName)
    Exit Sub

Errhandler:
    If Err = 53 Then
        Resume Next
    ElseIf Err = 62 Then
        MsgBox "Error in Loading Image"
        Exit Sub
    Else
        MsgBox "Error"

        SelfDefineIndex = 0
        'Unload frmDisplay
        ImgMin = MuRealMin
        ImgMax = MuRealMax
        Exit Sub
    End If


End Sub
```

mnuLowPassFilter_Click

Qualifiers: Private

```
Private Sub mnuLowPassFilter_Click()
    'Routine to do a low pass filter on the image

    'Global variables used
    '    - NV
    '    - Mu()
    '    - BmpFileName
    '    - BmpSize
    '    - BmpArray()

    Dim I As Integer, j As Integer
    Dim k As Integer, L As Integer
    Dim M As Integer, N As Integer
    Dim SumZ As Single
    Dim NewTop As Integer, NewBottom As Integer
    Dim NewLeft As Integer, NewRight As Integer
    Dim NewMu() As Single
    Dim Filter() As Single
    Dim FilterSize As Integer
    Dim HalfFilterSize As Integer
    Dim TotalFilterSize

    ReDim NewMu(1 To NV, 1 To NV)
    FilterSize = 3
    HalfFilterSize = Int(FilterSize / 2)
    TotalFilterSize = FilterSize * FilterSize
    ReDim Filter(1 To FilterSize, 1 To FilterSize)
    For I = 1 To FilterSize
        For j = 1 To FilterSize
            Filter(I, j) = 1
        Next j
    Next I

    NewTop = 1 + HalfFilterSize
    NewBottom = NV - HalfFilterSize
    NewLeft = 1 + HalfFilterSize
    NewRight = NV - HalfFilterSize

    'Now construct the new filtered array
    For I = NewLeft To NewRight
        Debug.Print I
        For j = NewTop To NewBottom
            SumZ = 0
```

```
            For k = (I - HalfFilterSize) To (I + HalfFilterSize)
                For L = (j - HalfFilterSize) To (j + HalfFilterSize)
                    'We should vary M and N from 1 To FilterSize
                    M = k + HalfFilterSize - I + 1
                    N = L + HalfFilterSize - j + 1
                    SumZ = SumZ + Mu(k, L) * Filter(M, N)
                Next L
            Next k
            NewMu(I, j) = SumZ / TotalFilterSize
        Next j
    Next I

    Call GenerateBitmapArray(NewMu(), NV, BmpArray(), BmpSize)
    'Now create the DIB file which will be read in
    Call WritePaletteAndBitmap(BmpFileName, BmpArray(), BmpSize)
    'Now copy NewMu into Mu
    For I = 1 To NV
        For j = 1 To NV
            Mu(I, j) = NewMu(I, j)
        Next j
    Next I

    Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
    imgView.Picture = LoadPicture(BmpFileName)

End Sub
```

mnuOption_Click

Qualifiers: Private

```
Private Sub mnuOption_Click()
    frmMinMax.Show

End Sub
```

mnuSizeRegionHelp_Click

Qualifiers: Private

```
Private Sub mnuSizeRegionHelp_Click()
    'FileMsg "DISPLAY.MSG", 5
End Sub
```

mnuSlideShow_Click

Qualifiers: Private

```
Private Sub mnuSlideShow_Click()
    'Save a complete BMP file. This will include writing both the Palette and the
    'Bitmap data

    'Global variables used
    '    - Palette()

    Dim StartTime, EndTime
    'Declarations for the DIB file name
    Dim I As Integer, j As Integer
    Dim PixelValue As String * 1
    Dim DibStartPos
    Dim DibFileNum, DibFileName
```

```
    'Other miscellaneous declarations
    Dim bmfh As BITMAPFILEHEADER
    Dim bmih As BITMAPINFOHEADER
    Dim rgbq As RGBQUAD

    Call ReadPaletteFile
    Call SetValuesForBMFH(bmfh)
    Call SetValuesForBMIH(bmih)

    StartTime = Timer
    DibStartPos = 1
    DibFileName = "d:\hsieh\cat\transmit\data\c3av\bmp\c3av3.bmp"
    DibFileNum = FreeFile
    Open DibFileName For Binary As DibFileNum
    Put #DibFileNum, DibStartPos, bmfh
    Put #DibFileNum, , bmih

    For I = 0 To 255
        rgbq.rgbBlue = Chr$(Palette(1, I))
        rgbq.rgbGreen = Chr$(Palette(2, I))
        rgbq.rgbRed = Chr$(Palette(3, I))
        Put #DibFileNum, , rgbq
    Next I

    'Now write the Actual Pixels
    For I = BmpSize To 1 Step -1
        For j = 1 To BmpSize
            PixelValue = Chr$(BmpArray(I, j))
            Put #DibFileNum, , PixelValue
        Next j
    Next I

    Close #DibFileNum
    'Debug.Print "Time Taken = "; Timer - StartTime
    'picView.ScaleWidth = 120 'NV
    'picView.ScaleHeight = 120'NV
    'PelsPerScaleWidth = picView.Width / NV
    'PelsPerScaleHeight = picView.Height / NV
    'imgView.Left = 0
    'imgView.Top = 0

    imgView.Picture = LoadPicture(BmpFileName)
    ImageVisible = True
End Sub
```

mnuStatisticsHelp_Click

Qualifiers: Private

```
Private Sub mnuStatisticsHelp_Click()
    'FileMsg "DISPLAY.MSG", 6
End Sub
```

mnuToggleRegionHelp_Click

Qualifiers: Private

```
Private Sub mnuToggleRegionHelp_Click()
    'FileMsg "DISPLAY.MSG", 4
End Sub
```

mnuToneHelp_Click

Qualifiers: Private

```
Private Sub mnuToneHelp_Click()
    'FileMsg "DISPLAY.MSG", 3
End Sub
```

mnuWhiteDown_Click

Qualifiers: Private

```
Private Sub mnuWhiteDown_Click()
    'Do a White Down

    'Global variables used
    '   - BmpFileName
    '   - Palette()
    '   - RegionXPos
    '   - RegionYPos
    '   - RegionRadius
    '   - WhiteThreshold

    Dim j

    If WhiteThreshold < 0 Then
        Exit Sub
    End If

    Debug.Print "WhiteThreshold = "; WhiteThreshold
    For j = 255 To WhiteThreshold Step -1
        Palette(1, j) = Palette(1, 255)
        Palette(2, j) = Palette(2, 255)
        Palette(3, j) = Palette(3, 255)
    Next j

    WritePalette (BmpFileName)
    Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)
    imgView.Picture = LoadPicture(BmpFileName)
    WhiteThreshold = WhiteThreshold - 10

End Sub
```

ReadPaletteFile

Qualifiers: Private

```
Private Sub ReadPaletteFile()
    'Read in the Color Palette

    'Global variables used
    '   - Palette()

    Dim I As Integer, j As Integer
    Dim Number
    Dim NumberString As String * 8
    Dim PalStartPos
    Dim PalFileNum
    Dim PalFileName As String

    ReDim Palette(1 To 3, 0 To 255)
    For I = 1 To 3
        For j = 0 To 255
            Palette(I, j) = 0
        Next j
    Next I
```

```
    PalFileNum = FreeFile
    'XXX - Change hardcoded file name later
    PalFileName = App.Path & "\" & "PALET.TXT"
    Open PalFileName For Binary As PalFileNum
    PalStartPos = 1

    For I = 1 To 3
        'XXX - Change the limit of J later
        For j = 0 To 246
            Get #PalFileNum, PalStartPos, NumberString

            Number = Val(NumberString)
            Palette(I, j + 9) = Number
            PalStartPos = PalStartPos + 8
            If (j + 1) Mod 9 = 0 Then
                PalStartPos = PalStartPos + 2
            End If
        Next j
        PalStartPos = PalStartPos + 2
    Next I

    Close #PalFileNum
End Sub
```

ReadRadonFile

Qualifiers:	Private		
Arguments:	RadonFile	String	By Value
	Mu	Single	By Ref.
	NV	Integer	By Ref.

```
Private Sub ReadRadonFile(ByVal RadonFile As String, Mu() As Single, NV As Integer)
    'Read in the results of the image reconstruction algorithm - it will be a .RAD
file

    'Global variables used
    '   None

    Dim I As Integer, j As Integer
    Dim Ix As Integer, Iy As Integer
    Dim FileNum, FileLength, NextLine
    Dim BufLen As Long

    BufLen = 120 * 120  'We provide a large buffer size
    FileNum = FreeFile
    Open RadonFile For Input As FileNum Len = BufLen

    Line Input #FileNum, NextLine
    NV = Val(NextLine)
    ReDim Mu(1 To NV, 1 To NV)

    For Iy = 1 To NV
        DoEvents
        For Ix = 1 To NV
            Line Input #FileNum, NextLine
            Mu(Ix, Iy) = Val(NextLine)
        Next Ix
    Next Iy
    Close #FileNum

End Sub
```

RecalculateCenterOfRegion

Qualifiers:	Private		
Arguments:	NewX	Variant	By Ref.

NewRadius	Variant	By Value

```
Private Sub RecalculateCenterOfRegion(NewX, NewY, ByVal NewRadius)
    'Routine used to recenter the circular region, in case it falls outside the image
area
    'For doing this we compare the size parameters of the region with the extent of
the
    'image area

    'Global variables used
    '   None

    Dim Offset

    If (NewX - NewRadius) < 0 Then
        NewX = NewRadius
    End If

    If (NewX + NewRadius) > picView.ScaleWidth Then
        NewX = picView.ScaleWidth - NewRadius
    End If

    If (NewY - NewRadius) < 0 Then
        NewY = NewRadius
    End If

    If (NewY + NewRadius) > picView.ScaleHeight Then
        NewY = picView.ScaleHeight - NewRadius
    End If
End Sub
```

ReduceRegion

Qualifiers: Private

```
Private Sub ReduceRegion()
    'Shrink the size of the region by a specified amount
    'Global variables used
    '   - RegionRadiusIncrement

    Dim NewRegionRadius

    Call EraseRegionAtOldPosition(RegionXPos, RegionYPos, RegionRadius)

    NewRegionRadius = RegionRadius - RegionRadiusIncrement
    If NewRegionRadius <= 0 Then
        Exit Sub
    Else
        RegionRadius = NewRegionRadius
    End If

    Call RecalculateCenterOfRegion(RegionXPos, RegionYPos, RegionRadius)
    Call DisplayRegionAtNewPosition(RegionXPos, RegionYPos, RegionRadius)
    Call ResetValuesInAllPanels(RegionXPos, RegionYPos, RegionRadius)
End Sub
```

ResetValuesInAllPanels

Qualifiers:	Private		
Arguments:	RegionXPos	Variant	By Ref.
	RegionYPos	Variant	By Ref.
	RegionRadius	Variant	By Ref.

```
Private Sub ResetValuesInAllPanels(RegionXPos, RegionYPos, RegionRadius)
```

```
    'Change the displayed values in the region extent display panels
    'Global variables used
    '   None

    Dim RegionLeft As Integer
    Dim RegionRight As Integer
    Dim RegionTop As Integer
    Dim RegionBottom As Integer

    Call FindRegionBoundaries(RegionXPos, RegionYPos, RegionRadius, RegionLeft,
RegionRight, RegionTop, RegionBottom)
    pnlRegionLeft.Caption = Format$(RegionLeft)
    pnlRegionRight.Caption = Format$(RegionRight)
    pnlRegionTop.Caption = Format$(RegionTop)
    pnlRegionBottom.Caption = Format$(RegionBottom)
End Sub
```

SetAllPanelsToEmptyString

Qualifiers: Private

```
Private Sub SetAllPanelsToEmptyString()
    'Clear all region extent display panels

    'Global variables used
    '   None

    pnlRegionLeft.Caption = ""
    pnlRegionRight.Caption = ""
    pnlRegionTop.Caption = ""
    pnlRegionBottom.Caption = ""
End Sub
```

SetValuesForBMFH

Qualifiers: Private
Arguments: bmfh BITMAPFILEHEADER By Ref.

```
Private Sub SetValuesForBMFH(bmfh As BITMAPFILEHEADER)
    'Set the default values for the Header block of a file of the BMP format

    'Global variables used
    '   - BmpSize
    '   - SizeOfBMFH
    '   - SizeOfBMIH
    '   - SizeOfColorTable
    '   - BmpImageSize

    Dim BmpImageSize As Long

    BmpImageSize = CLng(BmpSize) * CLng(BmpSize)
    bmfh.bfType = 19778 'The String "BM"
    bmfh.bfSize = SizeOfBMFH + SizeOfBMIH + SizeOfColorTable + BmpImageSize
    bmfh.bfReserved1 = 0
    bmfh.bfReserved2 = 0
    bmfh.bfOffBits = SizeOfBMFH + SizeOfBMIH + SizeOfColorTable
End Sub
```

SetValuesForBMIH

Qualifiers: Private

Arguments: bmih BITMAPINFOHEADER By Ref.

```
Private Sub SetValuesForBMIH(bmih As BITMAPINFOHEADER)
    'Set the default values for the Information block of a file of the BMP format,
this
    'is also part of the Header of the file

    'Global variables used
    '   - BmpSize
    '   - SizeOfBMIH
    '   - SizeOfColorTable
    '   - BmpImageSize
    '   - NumberOfColors

    Dim BmpImageSize As Long

    BmpImageSize = CLng(BmpSize) * CLng(BmpSize)
    bmih.biSize = SizeOfBMIH
    bmih.biWidth = BmpSize
    bmih.biHeight = BmpSize
    bmih.biPlanes = 1
    bmih.biBitCount = 8
    bmih.biCompression = 0
    bmih.biSizeImage = BmpImageSize

    bmih.biXPelsPerMeter = 0
    bmih.biYPelsPerMeter = 0
    bmih.biClrUsed = NumberOfColors
    bmih.biClrImportant = NumberOfColors
End Sub
```

spnEnlargeReduce_SpinDown

Qualifiers: Private

```
Private Sub spnEnlargeReduce_SpinDown()
    'Shrink the region

    'Global variables used
    '   None

    Call ReduceRegion
End Sub
```

spnEnlargeReduce_SpinUp

Qualifiers: Private

```
Private Sub spnEnlargeReduce_SpinUp()
    'Enlarge the region

    'Global variables used
    '   None

    Call EnlargeRegion
End Sub
```

Timer1_Timer

Qualifiers: Private

```
Private Sub Timer1_Timer()
    'The timer is called at repeatedly to refresh the screen
    'This refresh is necessary after an expose event has ocurred

    'Global variables used
    '    - RedrawRegion
    '    - RegionVisible
    '    - RegionXPos
    '    - RegionYPos
    '    - RegionRadius

    If (RedrawRegion = True) And (RegionVisible = True) Then
        picView.Cls
        Debug.Print "Redrawing region now"
        RedrawRegion = False
        Call RecalculateCenterOfRegion(RegionXPos, RegionYPos, RegionRadius)
        Call DisplayRegionAtNewPosition(RegionXPos, RegionYPos, RegionRadius)
        Call ResetValuesInAllPanels(RegionXPos, RegionYPos, RegionRadius)
    End If
End Sub
```

WritePalette

Qualifiers:	Private		
Arguments:	DibFileName	String	By Value

```
Private Sub WritePalette(ByVal DibFileName As String)
    'Write a midified palette to the palette file

    'Global variables used
    '    - Palette()

    Dim StartTime, EndTime
    'Declarations for the DIB file name
    Dim I As Integer, j As Integer
    Dim PixelValue As String * 1
    Dim DibStartPos
    Dim DibFileNum
    'Other miscellaneous declarations
    Dim bmfh As BITMAPFILEHEADER
    Dim bmih As BITMAPINFOHEADER
    Dim rgbq As RGBQUAD

    Call SetValuesForBMFH(bmfh)
    Call SetValuesForBMIH(bmih)

    StartTime = Timer
    DibStartPos = 1
    DibFileNum = FreeFile
    Open DibFileName For Binary As DibFileNum
    Put #DibFileNum, DibStartPos, bmfh
    Put #DibFileNum, , bmih

    For I = 0 To 255
        rgbq.rgbBlue = Chr$(Palette(1, I))
        rgbq.rgbGreen = Chr$(Palette(2, I))
        rgbq.rgbRed = Chr$(Palette(3, I))
        Put #DibFileNum, , rgbq
    Next I

    Close #DibFileNum
    Debug.Print "Time Taken = "; Timer - StartTime

End Sub
```

WritePaletteAndBitmap

Qualifiers:	Private		
Arguments:	DibFileName	String	By Value
	BmpArray	Integer	By Ref.
	BmpSize	Integer	By Value

```
Private Sub WritePaletteAndBitmap(ByVal DibFileName As String, BmpArray() As Integer,
ByVal BmpSize As Integer)
    'Save a complete BMP file. This will include writing both the Palette and the
    'Bitmap data

    'Global variables used
    '   - Palette()

    Dim StartTime, EndTime
    'Declarations for the DIB file name
    Dim I As Integer, j As Integer
    Dim PixelValue As String * 1
    Dim DibStartPos
    Dim DibFileNum
    'Other miscellaneous declarations
    Dim bmfh As BITMAPFILEHEADER
    Dim bmih As BITMAPINFOHEADER
    Dim rgbq As RGBQUAD

    Call ReadPaletteFile
    Call SetValuesForBMFH(bmfh)
    Call SetValuesForBMIH(bmih)

    StartTime = Timer
    DibStartPos = 1
    DibFileNum = FreeFile
    Open DibFileName For Binary As DibFileNum
    Put #DibFileNum, DibStartPos, bmfh
    Put #DibFileNum, , bmih

    For I = 0 To 255
        rgbq.rgbBlue = Chr$(Palette(1, I))
        rgbq.rgbGreen = Chr$(Palette(2, I))
        rgbq.rgbRed = Chr$(Palette(3, I))
        Put #DibFileNum, , rgbq
    Next I

    'Now write the Actual Pixels
    For I = BmpSize To 1 Step -1
        For j = 1 To BmpSize
            PixelValue = Chr$(BmpArray(I, j))
            Put #DibFileNum, , PixelValue
        Next j
    Next I

    Close #DibFileNum
    Debug.Print "Time Taken = "; Timer - StartTime

End Sub
```

CompPercent

Qualifiers:	Public		
Arguments:	InArr	Single	By Ref.
	NV	Integer	By Value
	DoloPer	Variant	By Ref.
	GypPer	Variant	By Ref.
	VoidPer	Single	By Ref.

```
Sub CompPercent(InArr() As Single, ByVal NV As Integer, DoloPer, GypPer, VoidPer As
Single)
```

```
    Dim IndexGypsum As Integer
    Dim IndexDolomite As Integer
    Dim IndexVoid As Integer
    Dim j, k, index As Long
    Dim TotalVoxel As Single

    IndexGypsum = 0
    IndexDolomite = 0
    IndexVoid = 0
    index = 0

    For j = 1 To NV
        For k = 1 To NV
            If InArr(k, j) = 1# Then
                IndexVoid = IndexVoid + 1
            ElseIf InArr(k, j) = 2# Then
                IndexGypsum = IndexGypsum + 1
            ElseIf InArr(k, j) = 3# Then
                IndexDolomite = IndexDolomite + 1
            Else
                index = index + 1
            End If
        Next k
    Next j

    TotalVoxel = (IndexVoid + IndexGypsum + IndexDolomite + index) * 1#
    DoloPer = IndexDolomite / TotalVoxel
    GypPer = IndexGypsum / TotalVoxel
    VoidPer = IndexVoid / TotalVoxel

End Sub
```

Functions

Maximum

Qualifiers:	Private		
Arguments:	X	Variant	By Ref.
	Y	Variant	By Ref.
Returns:	Variant		

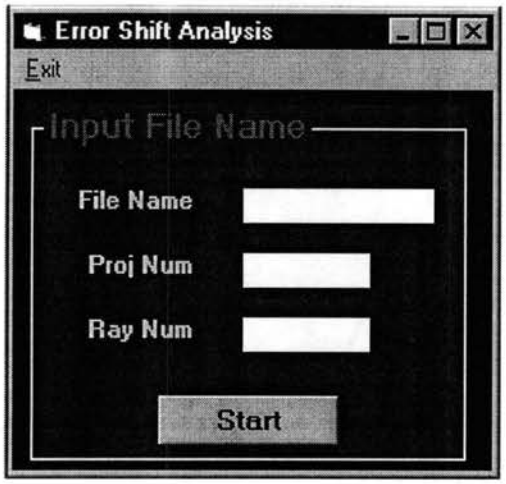
```
Private Function Maximum(X, Y) As Variant
    'Find the maximum of the two arguments

    'Global variables used
    '   None

    If X > Y Then
        Maximum = X
    Else
        Maximum = Y
    End If
End Function
```

ERRSHIFT.FRM

Mod Date Fri Oct 04 12:45:16 1996
Size 7738



Declarations

```
Attribute VB_Name = "frmErrorShift"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
```

Menu

Caption	Shortcut	Name
&Exit		mnuExit

Subroutines

CmdErrorShift_Click

Qualifiers: Private

```
Private Sub CmdErrorShift_Click()
    Dim I, StrLen, TempString

    If InputFile = "" Then
        Beep
        MsgBox "Incorrect Input file name"
        Exit Sub
    End If

    '**** Check whether the Input File has the extension .DAT ****
    StrLen = Len(InputFile)
    Debug.Print InputFile
    If UCase$(Mid$(InputFile, StrLen - 3, 4)) <> ".DAT" Then
        Beep
```

```
        MsgBox "Error: Input file name must end in .DAT"
        Exit Sub
    End If

    Proj = Val(txtProj.Text)
    Ray = Val(txtRay.Text)

    '**** Assign Variable Values For Now ****

    If Proj = 0 Or Ray = 0 Then
        Beep
        MsgBox "Incorrect Input"
        Exit Sub
    End If

    'read input file (can be independent from .frm)
    Dim j As Integer, k As Integer
    Dim ArrNum As Integer

    ArrNum = 92

    Dim AirCount1 As Long, AirCount2 As Long
    Dim InputFileNum, FileLength, NextLine
    Dim ErrorProj() As Single
    ReDim ErrorProj(1 To Proj, 1 To Ray) As Single

    Dim ErrorMu() As Single
    ReDim ErrorMu(1 To ArrNum, 1 To ArrNum) As Single

    FileLength = Proj * (Ray + 2)
    InputFileNum = FreeFile
    Open InputFile For Input As InputFileNum Len = FileLength

    For j = 1 To Proj
        DoEvents
        Line Input #InputFileNum, NextLine
        AirCount1 = Val(NextLine)
        Line Input #InputFileNum, NextLine
        AirCount2 = Val(NextLine)
        For k = 1 To Ray
            Line Input #InputFileNum, NextLine
            'If Val(NextLine) < .1 Then
            '    ErrorProj(j, k) = 0
            'Else
            '    ErrorProj(j, k) = 1
            'End If
            ErrorProj(j, k) = Val(NextLine)
        Next k
    Next j
    Close #InputFileNum

    Dim FileNum
    OutputFile = "d:\hsieh\cat\transmit\data\c2av15m\error.rad"
    FileNum = FreeFile
    Open OutputFile For Output As #FileNum

    Print #FileNum, ArrNum
    For j = 1 To ArrNum
        For k = 1 To ArrNum
            If j > 1 And j < 92 And k > 1 And k < 92 Then
                ErrorMu(j, k) = ErrorProj(j - 1, k - 1)
                Print #FileNum, ErrorMu(j, k)
            Else
                ErrorMu(j, k) = 2
                Print #FileNum, ErrorMu(j, k)
            End If
        Next k
    Next j
    Close #FileNum


    MsgBox "Analysis Completed!"

End Sub
```

mnuExit_Click

Qualifiers: Private

```
Private Sub mnuExit_Click()
    frmErrorShift.Hide
    frmRadon.Show
    Unload frmErrorShift

End Sub
```

TxtErrorShift_DblClick

Qualifiers: Private

```
Private Sub TxtErrorShift_DblClick()
    Dim temp$

    frmGetFile.Caption = "Input File Name"
    frmGetFile.FileTypes.AddItem "All Files (*.*)"
    frmGetFile.Show MODAL
    temp$ = frmGetFile.FullPath.Text
    If temp$ <> "" Then
        txtErrorShift.Text = UCase$(frmGetFile.Text1)
        'We set a global variable here
        InputFile = UCase$(frmGetFile.FullPath.Text)
    End If

End Sub
```

TxtErrorShift_GotFocus

Qualifiers: Private

```
Private Sub TxtErrorShift_GotFocus()
    txtErrorShift.SelStart = 0
    txtErrorShift.SelLength = 65000

End Sub
```

TxtProj_GotFocus

Qualifiers: Private

```
Private Sub TxtProj_GotFocus()
    txtProj.SelStart = 0
    txtProj.SelLength = 65000

End Sub
```

TxtRay_GotFocus

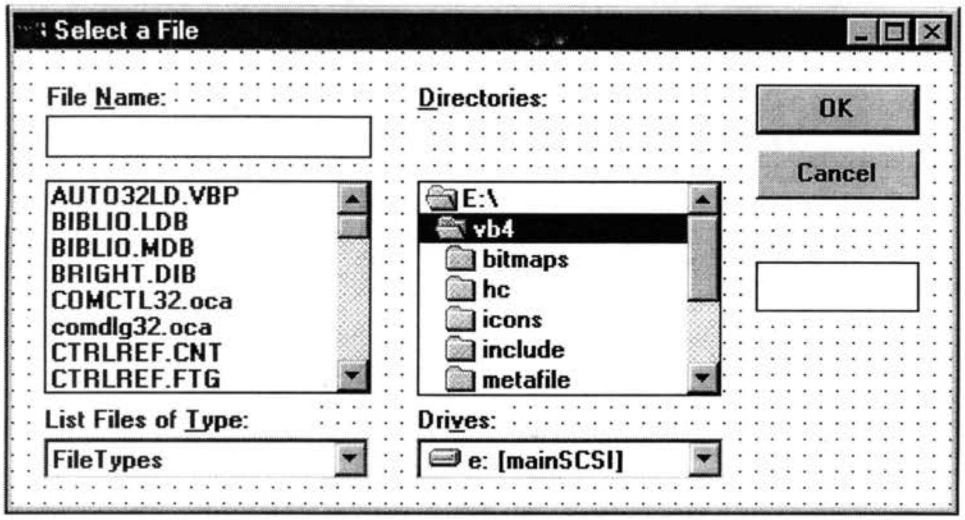
Qualifiers: Private

```
Private Sub TxtRay_GotFocus()
    txtRay.SelStart = 0
    txtRay.SelLength = 65000

End Sub
```

GETFILE.FRM

Mod Date	Fri Oct 04 12:45:15 1996
Size	9993



Declarations

```
Attribute VB_Name = "frmGetFile"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
'Declarations for GETFILE.FRM

Const TEXTFLAG = 0
Const FILEFLAG = 1
Const DIRFLAG = 2
Dim SelectFlag As Integer
```

Subroutines

Command1_Click

Qualifiers: Private

```
Private Sub Command1_Click()
    'OK button; some errors can happen
    On Error GoTo ErrorTrap

    'Was the last change to the filename in Text1?
    If SelectFlag = TEXTFLAG Then
        File1.FileName = Text1.Text

        'We're done if FullPath was set
        If FullPath <> "" Then
            On Error GoTo 0
            ExitForm
        End If
```

```
        'Update directory list
        Dir1.Path = File1.Path

    'Was user only selecting a new directory?
    ElseIf SelectFlag = DIRFLAG Then
        Dir1.Path = Dir1.List(Dir1.ListIndex)
        Dir1_Change

    'Set FullPath to selected file
    Else
        If Right$(Dir1.Path, 1) = "\" Then
            FullPath.Text = Dir1.Path + Text1.Text
        Else
            FullPath.Text = Dir1.Path + "\" + Text1.Text
        End If

        'All done
        ExitForm
    End If

    Exit Sub

ErrorTrap:
    Beep
    Resume Next
End Sub
```

Command2_Click

Qualifiers: Private

```
Private Sub Command2_Click()
    'Cancel button; indicate by erasing FullPath
    FullPath = ""

    'All done
    ExitForm
End Sub
```

Dir1_Change

Qualifiers: Private

```
Private Sub Dir1_Change()
    'User selected new subdirectory
    FillLabel1

    'Update filename
    File1.FileName = Dir1.Path + "\" + File1.Pattern
    File1.Pattern = GetFileType$()

    'Update drive list
    Drive1.Drive = Dir1.Path

    'Update name of file
    Text1.Text = File1.Pattern

    'Set last change to directory
    SelectFlag = DIRFLAG
End Sub
```

Dir1_Click

Qualifiers: Private

```
Private Sub Dir1_Click()
    'User clicked on new subdirectory
    SelectFlag = DIRFLAG
End Sub
```

Drive1_Change

Qualifiers: Private

```
Private Sub Drive1_Change()
    'User changed drive; update directory
    Dir1.Path = Drive1.Drive

    'Display current pattern
    Text1.Text = File1.Pattern

    'Set last change to directory
    SelectFlag = DIRFLAG
End Sub
```

ExitForm

Qualifiers: Private

```
Private Sub ExitForm()
    'User might want different patterns next time
    FileTypes.Clear

    'Don't unload, simply hide
    frmGetFile.Hide
End Sub
```

File1_Click

Qualifiers: Private

```
Private Sub File1_Click()
    'User clicked on new filename
    Text1.Text = File1.FileName

    'Set last change to filename
    SelectFlag = FILEFLAG
End Sub
```

File1_DblClick

Qualifiers: Private

```
Private Sub File1_DblClick()
    'User double-clicked on a filename
    Command1_Click
End Sub
```

FileTypes_Click

Qualifiers: Private

```
Private Sub FileTypes_Click()
    'User selected new pattern from combo box
    File1.Pattern = GetFileType$()

    'Display pattern until a file is selected
    Text1.Text = File1.Pattern
End Sub
```

FillLabel1

Qualifiers: Private

```
Private Sub FillLabel1()
    'Display directory part of path
    Label1.Caption = Dir1.Path

    'If directory string is too long, squish it down
    If Label1.Width > 2200 Then

        'Extract drive part
        A$ = Left$(Dir1.Path, 3)
        B$ = Mid$(Dir1.Path, 4)

        'Extract last subdirectory part
        Do While InStr(B$, "\")
            B$ = Mid$(B$, InStr(B$, "\") + 1)
        Loop

        'Squish out middle part
        Label1.Caption = A$ + "...\" + B$
    End If
End Sub
```

Form_Activate

Qualifiers: Private

```
Private Sub Form_Activate()
    'Don't select any filename at first
    File1.ListIndex = -1

    'If no pattern list, default to *.*
    If FileTypes.ListCount = 0 Then
        FileTypes.AddItem "All Files (*.*)"
    End If

    'Default to first pattern in list
    FileTypes.ListIndex = 0

    'If no previous path, use application's path
    If FullPath.Text = "" Then
        FullPath.Text = App.Path + "\"
    End If

    'Update lists and labels
    File1.Pattern = GetFileType$()
    Text1.Text = File1.Pattern
```

```
    Dir1.Path = File1.Path
    FillLabel1
    SelectFlag = DIRFLAG
    FullPath = ""
End Sub
```

Form_KeyUp

Qualifiers:	Private		
Arguments:	KeyCode	Integer	By Ref.
	Shift	Integer	By Ref.

```
Private Sub Form_KeyUp(KeyCode As Integer, Shift As Integer)
    'Watch only for Alt plus N, D, T or V key
    If Shift = 4 Then
        Select Case KeyCode

        'Alt+N
        Case 78
            Text1.SetFocus

        'Alt+D
        Case 68
            Dir1.SetFocus

        'Alt+T
        Case 84
            FileTypes.SetFocus

        'Alt+V
        Case 86
            Drive1.SetFocus

        End Select
    End If
End Sub
```

Form_Load

Qualifiers: Private

```
Private Sub Form_Load()
    'Center form on screen
    frmGetFile.Left = (Screen.Width - frmGetFile.Width) / 2
    frmGetFile.Top = (Screen.Height - frmGetFile.Height) / 2
End Sub
```

Text1_Change

Qualifiers: Private

```
Private Sub Text1_Change()
    'Set last change to File Name field
    SelectFlag = TEXTFLAG
End Sub
```

Functions

GetFileType

Qualifiers: Private
Returns: String

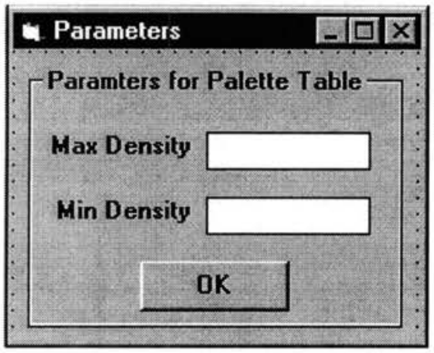
```
Private Function GetFileType$()
    'Get pattern description from combo box
    Tmp$ = FileTypes.Text

    'Find position of parentheses
    p1 = InStr(Tmp$, "(") + 1
    p2 = InStr(Tmp$, ")")

    'Return part between parentheses
    If p1 > 0 And p2 > p1 Then
        GetFileType$ = LCase$(Mid$(Tmp$, p1, p2 - p1))
    Else
        GetFileType$ = "*.*"
    End If
End Function
```

MINMAX.FRM

Mod Date Fri Oct 04 12:45:16 1996
Size 3929



Declarations

```
Attribute VB_Name = "frmMinMax"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
```

Subroutines

CmdMinMax_Click

Qualifiers: Private

```
Private Sub CmdMinMax_Click()
'===========================================================
    '6-29-96 hsieh added
    '       - define the same color palette Max and Min
    '         for all images, if you define them, otherwise
    '         the calculated min and max will be applied.
    ImgMin = Val(txtImgMin.Text)
    ImgMax = Val(txtImgMax.Text)
'===========================================================
    If ImgMin > ImgMax Then
        Beep
        MsgBox "Incorrect Input"
        Exit Sub
    End If

    If txtImgMin = "" Or txtImgMax = "" Then
        SelfDefineIndex = 0
    Else
        SelfDefineIndex = 1
    End If

    frmMinMax.Hide
    Unload frmMinMax


End Sub
```

TxtImgMax_GotFocus

Qualifiers: Private

```
Private Sub TxtImgMax_GotFocus()
    txtImgMax.SelStart = 0
    txtImgMax.SelLength = 65000
End Sub
```

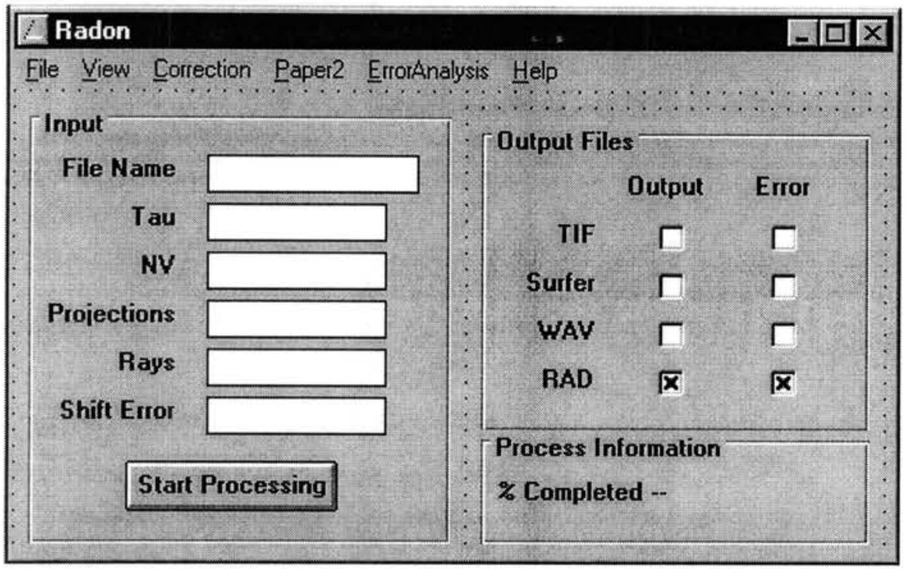
TxtImgMin_GotFocus

Qualifiers: Private

```
Private Sub TxtImgMin_GotFocus()
    txtImgMin.SelStart = 0
    txtImgMin.SelLength = 65000
End Sub
```

RADON.FRM

Mod Date	Wed Feb 05 14:33:12 1997
Size	23495



Declarations

```
Attribute VB_Name = "frmRadon"
Attribute VB_Creatable = False
Attribute VB_Exposed = False
Option Explicit
Dim ProcessInProgress As Integer
```

Menu

Caption	Shortcut	Name
&File		mnuFile
E&xit		mnuExit
&View		mnuView
&Correction		mnuCorrect
&Paper2		mnuSCDP
&ErrorAnalysis		mnuErrorAna
&Help		mnuHelp

Subroutines

cmdStart_Click

Qualifiers: Private

```
Private Sub cmdStart_Click()
    Dim I, StrLen, TempString

    If ProcessInProgress Then
        Beep
        MsgBox "A Radon Transform Calculation is already running"
        Exit Sub
    End If

    'We dont need the following statement since the string
    'InputFile will be set by the GetFile.frm
    'InputFile = txtInputFile.Text
    Tau = Val(txtTau.Text)
    NV = Val(txtNV.Text)
    ProjNum = Val(txtProjNum.Text)
    RayNum = Val(txtRayNum.Text)
    ShiftError = Val(txtShiftError.Text)

    If InputFile = "" Then
        Beep
        MsgBox "Incorrect Input file name"
        Exit Sub
    End If

    '**** Check whether the Input File has the extension .DAT ****
    StrLen = Len(InputFile)
    Debug.Print InputFile
    If UCase$(Mid$(InputFile, StrLen - 3, 4)) <> ".DAT" Then
        Beep
        MsgBox "Error: Input file name must end in .DAT"
        Exit Sub
    End If

    '**** Assign Variable Values For Now ****

    If Tau = 0 Or NV = 0 Or ProjNum = 0 Or RayNum = 0 Then
        Beep
        MsgBox "Incorrect Input"
        Exit Sub
    End If

    ProcessInProgress = True
    fraProcessInformation.Visible = True
    Call StartProcessing
    ProcessInProgress = False
    fraProcessInformation.Visible = False

End Sub
```

Form_Load

Qualifiers: Private

```
Private Sub Form_Load()
    ProcessInProgress = False
    fraProcessInformation.Visible = False
    'WorkingDirectory = Environ$("RADON")
    'If WorkingDirectory = "" Then
    '    MsgBox "Please set the Enviroment variable  - RADON -"
    'End If

    txtTau.Text = 0.5
    txtNV.Text = 120
    txtProjNum.Text = 90
    txtRayNum.Text = 90
    txtShiftError.Text = 1.1

End Sub
```

Form_Unload

Qualifiers: Private
Arguments: Cancel Integer By Ref.

```
Private Sub Form_Unload(Cancel As Integer)
    End
End Sub
```

mnuCorrect_Click

Qualifiers: Private

```
Private Sub mnuCorrect_Click()

    frmRadon.Hide
    frmCorrect.Show

    frmCorrect.txtTau.Text = 0.5
    frmCorrect.txtProjNum.Text = 90
    frmCorrect.txtRayNum.Text = 90
    frmCorrect.txtatten.Text = 0.0098
    frmCorrect.txtOD.Text = 57.06
    frmCorrect.txtID.Text = 51.18
    frmCorrect.txtShiftError.Text = 1.1
End Sub
```

mnuErrorAna_Click

Qualifiers: Private

```
Private Sub mnuErrorAna_Click()
    frmErrorShift.Show
    frmRadon.Hide

End Sub
```

mnuExit_Click

Qualifiers: Private

```
Private Sub mnuExit_Click()
    Unload frmRadon
    End
End Sub
```

mnuHelp_Click

Qualifiers: Private

```
Private Sub mnuHelp_Click()
    Dim Msg$
```

```
    Msg$ = "Help: Currently not supported"
    MsgBox Msg$
End Sub
```

mnuSCDP_Click

Qualifiers: Private

```
Private Sub mnuSCDP_Click()
   frmRadon.Hide
    FrmSpatial.Show

    FrmSpatial.txtScaling.Text = 1#
    FrmSpatial.txtStdDevNum.Text = 1
    FrmSpatial.TxtNumCompt.Text = 0#

    FrmSpatial.OptExp.Value = True
    FrmSpatial.OptSobel.Value = True
    FrmSpatial.txtEdgeThres.Text = 0.014

    FrmSpatial.txtMaxIntense.Text = 0.01
    FrmSpatial.txtMinIntense.Text = 0#
    FrmSpatial.txtScaleFactor.Text = 1#
    FrmSpatial.txtClassNum.Text = 150
    FrmSpatial.txtX.Text = 60
    FrmSpatial.txtY.Text = 60
    FrmSpatial.txtRadius.Text = 50#

    FrmSpatial.txtOriginNV.Text = 120
    FrmSpatial.txtNewNV.Text = 68

    NumberOfComponents = 0
    CompNums = 1
    FrmSpatial.txtCompNums.Text = 1#
    FrmSpatial.TxtScales.Text = 1#

  If FrmSpatial.ChkHisto.Value = 0 Then
    FrmSpatial.FraHisto.Enabled = False
    FrmSpatial.txtMaxIntense.Enabled = False
    FrmSpatial.txtMinIntense.Enabled = False
    FrmSpatial.LblMaxIntense.Enabled = False
    FrmSpatial.LblMinIntense.Enabled = False
    FrmSpatial.LblScaleFactor.Enabled = False
    FrmSpatial.LblClassNum.Enabled = False
    FrmSpatial.txtScaleFactor.Enabled = False
    FrmSpatial.txtClassNum.Enabled = False
    FrmSpatial.txtX.Enabled = False
    FrmSpatial.txtY.Enabled = False
    FrmSpatial.txtRadius.Enabled = False
    FrmSpatial.LblX.Enabled = False
    FrmSpatial.LblY.Enabled = False
    FrmSpatial.LblRadius.Enabled = False

ElseIf FrmSpatial.ChkHisto.Value = 1 Then
    FrmSpatial.FraHisto.Enabled = True
    FrmSpatial.txtMaxIntense.Enabled = True
    FrmSpatial.txtMinIntense.Enabled = True
    FrmSpatial.LblMaxIntense.Enabled = True
    FrmSpatial.LblMinIntense.Enabled = True
    FrmSpatial.LblScaleFactor.Enabled = True
    FrmSpatial.txtScaleFactor.Enabled = True
    FrmSpatial.LblClassNum.Enabled = True
    FrmSpatial.txtClassNum.Enabled = True
    FrmSpatial.txtX.Enabled = True
    FrmSpatial.txtY.Enabled = True
    FrmSpatial.txtRadius.Enabled = True
    FrmSpatial.LblX.Enabled = True
    FrmSpatial.LblY.Enabled = True
    FrmSpatial.LblRadius.Enabled = True
End If
```

```
If FrmSpatial.ChkDirect.Value = 0 Then
    FrmSpatial.FraDirect.Enabled = False
    FrmSpatial.txtCompNums.Enabled = False
    FrmSpatial.LblCompNums.Enabled = False
    FrmSpatial.TxtScales.Enabled = False
    FrmSpatial.LblScales.Enabled = False
    FrmSpatial.ComboCompNums.Enabled = False
    FrmSpatial.TxtThreshold.Enabled = False
    FrmSpatial.LblThreshold.Enabled = False

ElseIf FrmSpatial.ChkDirect.Value = 1 Then
    FrmSpatial.FraDirect.Enabled = True
    FrmSpatial.txtCompNums.Enabled = True
    FrmSpatial.LblCompNums.Enabled = True
    FrmSpatial.TxtScales.Enabled = True
    FrmSpatial.LblScales.Enabled = True
    FrmSpatial.ComboCompNums.Enabled = True
    FrmSpatial.TxtThreshold.Enabled = True
    FrmSpatial.LblThreshold.Enabled = True
End If

If FrmSpatial.ChkSampling.Value = 0 Then
    FrmSpatial.fraSampling.Enabled = False
    FrmSpatial.txtOriginNV.Enabled = False
    FrmSpatial.txtNewNV.Enabled = False
    FrmSpatial.LblOriginNV.Enabled = False
    FrmSpatial.LblNewNV.Enabled = False
ElseIf FrmSpatial.ChkSampling.Value = 1 Then
    FrmSpatial.fraSampling.Enabled = True
    FrmSpatial.txtOriginNV.Enabled = True
    FrmSpatial.txtNewNV.Enabled = True
    FrmSpatial.LblOriginNV.Enabled = True
    FrmSpatial.LblNewNV.Enabled = True
End If

If FrmSpatial.ChkSpatial.Value = 0 Then
    FrmSpatial.FraSpatial.Enabled = False
    FrmSpatial.framData.Enabled = False
    FrmSpatial.framEdge.Enabled = False
    FrmSpatial.framComp.Enabled = False

    FrmSpatial.LblNumCompt.Enabled = False
    FrmSpatial.TxtNumCompt.Enabled = False
    FrmSpatial.LblScaling.Enabled = False
    FrmSpatial.txtScaling.Enabled = False
    FrmSpatial.LblStdDevNum.Enabled = False
    FrmSpatial.txtStdDevNum.Enabled = False
    FrmSpatial.OptGaussian.Enabled = False
    FrmSpatial.OptGauThesh.Enabled = False
    FrmSpatial.ComboNumCompt.Enabled = False
    FrmSpatial.lblMean.Enabled = False
    FrmSpatial.TxtMean.Enabled = False
    FrmSpatial.LblStd.Enabled = False
    FrmSpatial.TxtStd.Enabled = False

    FrmSpatial.OptLinear.Enabled = False
    FrmSpatial.OptExp.Enabled = False

    FrmSpatial.OptLaplacian.Enabled = False
    FrmSpatial.OptRoberts.Enabled = False
    FrmSpatial.OptSobel.Enabled = False
    FrmSpatial.lblEdgeThres.Enabled = False
    FrmSpatial.txtEdgeThres.Enabled = False

ElseIf FrmSpatial.ChkSpatial.Value = 1 Then
    FrmSpatial.FraSpatial.Enabled = True
    FrmSpatial.framData.Enabled = True
    FrmSpatial.framEdge.Enabled = True
    FrmSpatial.framComp.Enabled = True

    FrmSpatial.LblNumCompt.Enabled = True
    FrmSpatial.TxtNumCompt.Enabled = True
    FrmSpatial.LblScaling.Enabled = True
    FrmSpatial.txtScaling.Enabled = True
    FrmSpatial.LblStdDevNum.Enabled = True
    FrmSpatial.txtStdDevNum.Enabled = True
```

```
    FrmSpatial.OptGaussian.Enabled = True
    FrmSpatial.OptGauThesh.Enabled = True
    FrmSpatial.ComboNumCompt.Enabled = True
    FrmSpatial.lblMean.Enabled = True
    FrmSpatial.TxtMean.Enabled = True
    FrmSpatial.LblStd.Enabled = True
    FrmSpatial.TxtStd.Enabled = True

    FrmSpatial.OptLinear.Enabled = True
    FrmSpatial.OptExp.Enabled = True

    FrmSpatial.OptRoberts.Enabled = True
    FrmSpatial.OptSobel.Enabled = True
    FrmSpatial.lblEdgeThres.Enabled = True
    FrmSpatial.txtEdgeThres.Enabled = True
End If

End Sub
```

mnuView_Click

Qualifiers: Private

```
Private Sub mnuView_Click()
    frmRadon.Hide
    frmDisplay.Show
End Sub
```

txtInputFile_DblClick

Qualifiers: Private

```
Private Sub txtInputFile_DblClick()
    Dim temp$

    frmGetFile.Caption = "Input File Name"
    frmGetFile.FileTypes.AddItem "All Files (*.*)"
    frmGetFile.Show MODAL
    temp$ = frmGetFile.FullPath.Text
    If temp$ <> "" Then
        txtInputFile.Text = UCase$(frmGetFile.Text1)
        'We set a global variable here
        InputFile = UCase$(frmGetFile.FullPath.Text)
    End If

End Sub
```

txtInputFile_GotFocus

Qualifiers: Private

```
Private Sub txtInputFile_GotFocus()
    txtInputFile.SelStart = 0
    txtInputFile.SelLength = 65000
End Sub
```

txtNV_GotFocus

Qualifiers: Private

```
Private Sub txtNV_GotFocus()
    txtNV.SelStart = 0
    txtNV.SelLength = 65000
End Sub
```

TxtProjNum_GotFocus

Qualifiers: Private

```
Private Sub TxtProjNum_GotFocus()
    txtProjNum.SelStart = 0
    txtProjNum.SelLength = 65000
End Sub
```

TxtRayNum_GotFocus

Qualifiers: Private

```
Private Sub TxtRayNum_GotFocus()
    txtRayNum.SelStart = 0
    txtRayNum.SelLength = 65000
End Sub
```

txtShiftError_GotFocus

Qualifiers: Private

```
Private Sub txtShiftError_GotFocus()
    txtShiftError.SelStart = 0
    txtShiftError.SelLength = 65000
End Sub
```

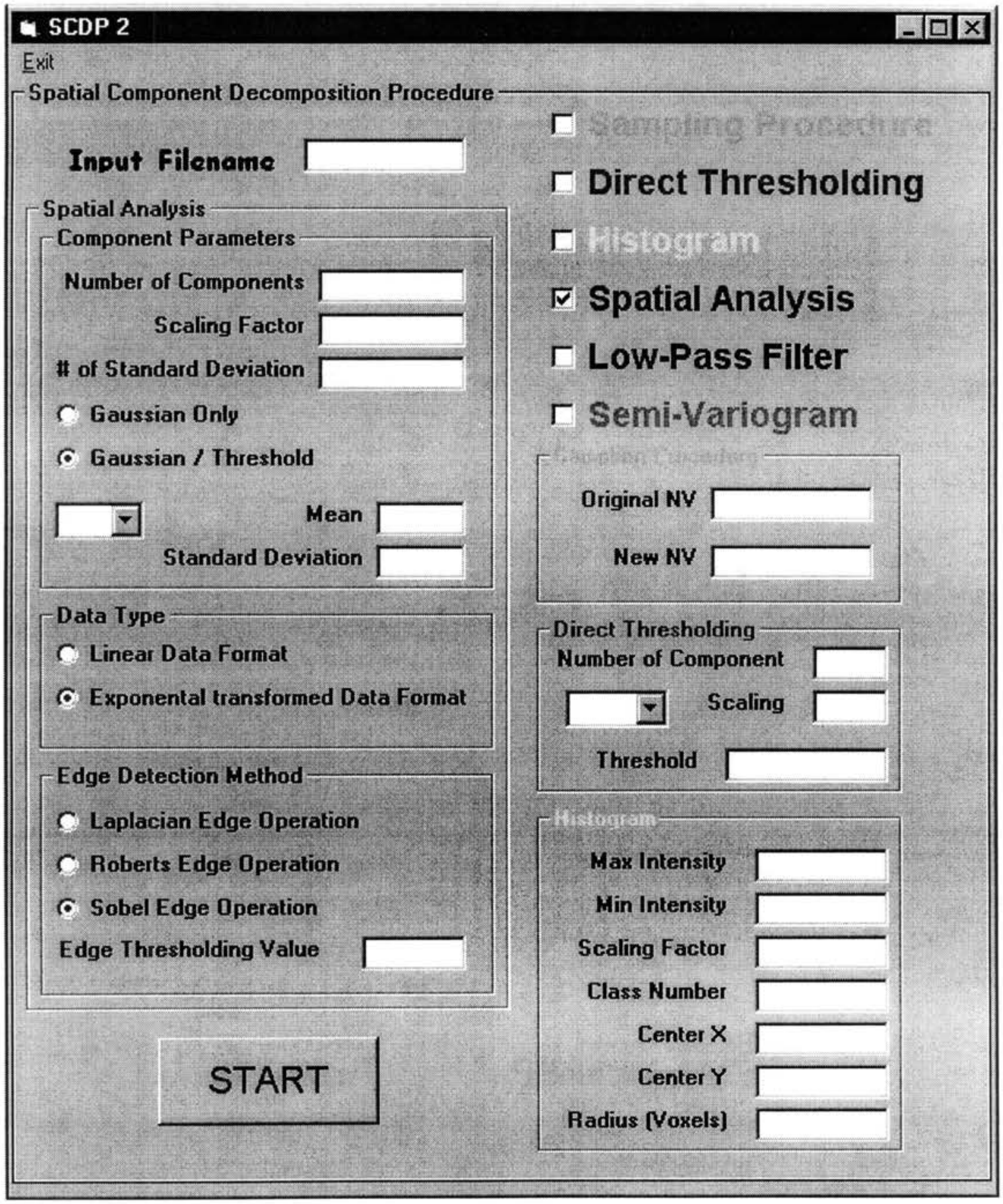
TxtTau_GotFocus

Qualifiers: Private

```
Private Sub TxtTau_GotFocus()
    txtTau.SelStart = 0
    txtTau.SelLength = 65000
End Sub
```

SPATIAL.FRM

Mod Date	Tue Sep 02 13:59:29 1997
Size	34540



Declarations

```
Attribute VB_Name = "FrmSpatial"
```

```
Attribute VB_Creatable = False
Attribute VB_Exposed = False
```

Menu

Caption	Shortcut	Name
&Exit		mnuExit

Subroutines

ChkDirect_Click

Qualifiers: Private

```
Private Sub ChkDirect_Click()

If FrmSpatial.ChkDirect.Value = 0 Then
    FrmSpatial.FraDirect.Enabled = False
    FrmSpatial.txtCompNums.Enabled = False
    FrmSpatial.LblCompNums.Enabled = False
    FrmSpatial.TxtScales.Enabled = False
    FrmSpatial.LblScales.Enabled = False
    FrmSpatial.ComboCompNums.Enabled = False
    FrmSpatial.TxtThreshold.Enabled = False
    FrmSpatial.LblThreshold.Enabled = False

ElseIf FrmSpatial.ChkDirect.Value = 1 Then
    FrmSpatial.FraDirect.Enabled = True
    FrmSpatial.txtCompNums.Enabled = True
    FrmSpatial.LblCompNums.Enabled = True
    FrmSpatial.TxtScales.Enabled = True
    FrmSpatial.LblScales.Enabled = True
    FrmSpatial.ComboCompNums.Enabled = True
    FrmSpatial.TxtThreshold.Enabled = True
    FrmSpatial.LblThreshold.Enabled = True
End If

End Sub
```

ChkHisto_Click

Qualifiers: Private

```
Private Sub ChkHisto_Click()
If ChkHisto.Value = 0 Then
    FrmSpatial.FraHisto.Enabled = False
    FrmSpatial.txtMaxIntense.Enabled = False
    FrmSpatial.txtMinIntense.Enabled = False
    FrmSpatial.LblMaxIntense.Enabled = False
    FrmSpatial.LblMinIntense.Enabled = False
    FrmSpatial.LblScaleFactor.Enabled = False
    FrmSpatial.LblClassNum.Enabled = False
    FrmSpatial.txtScaleFactor.Enabled = False
    FrmSpatial.txtClassNum.Enabled = False
    FrmSpatial.txtX.Enabled = False
    FrmSpatial.txtY.Enabled = False
    FrmSpatial.txtRadius.Enabled = False
    FrmSpatial.lblX.Enabled = False
    FrmSpatial.lblY.Enabled = False
    FrmSpatial.LblRadius.Enabled = False

ElseIf ChkHisto.Value = 1 Then
```

```
    FrmSpatial.FraHisto.Enabled = True
    FrmSpatial.txtMaxIntense.Enabled = True
    FrmSpatial.txtMinIntense.Enabled = True
    FrmSpatial.LblMaxIntense.Enabled = True
    FrmSpatial.LblMinIntense.Enabled = True
    FrmSpatial.LblScaleFactor.Enabled = True
    FrmSpatial.txtScaleFactor.Enabled = True
    FrmSpatial.LblClassNum.Enabled = True
    FrmSpatial.txtClassNum.Enabled = True
    FrmSpatial.txtX.Enabled = True
    FrmSpatial.txtY.Enabled = True
    FrmSpatial.txtRadius.Enabled = True
    FrmSpatial.lblX.Enabled = True
    FrmSpatial.lblY.Enabled = True
    FrmSpatial.LblRadius.Enabled = True

End If

End Sub
```

ChkLowPass_Click

Qualifiers: Private

```
Private Sub ChkLowPass_Click()
If ChkLowPass.Value = 0 Then

ElseIf ChkLowPass.Value = 1 Then

End If

End Sub
```

ChkSampling_Click

Qualifiers: Private

```
Private Sub ChkSampling_Click()
If ChkSampling.Value = 0 Then
    FrmSpatial.fraSampling.Enabled = False
    FrmSpatial.txtOriginNV.Enabled = False
    FrmSpatial.txtNewNV.Enabled = False
    FrmSpatial.LblOriginNV.Enabled = False
    FrmSpatial.LblNewNV.Enabled = False
ElseIf ChkSampling.Value = 1 Then
    FrmSpatial.fraSampling.Enabled = True
    FrmSpatial.txtOriginNV.Enabled = True
    FrmSpatial.txtNewNV.Enabled = True
    FrmSpatial.LblOriginNV.Enabled = True
    FrmSpatial.LblNewNV.Enabled = True
End If

End Sub
```

ChkSpatial_Click

Qualifiers: Private

```
Private Sub ChkSpatial_Click()
If ChkSpatial.Value = 0 Then
    FrmSpatial.FraSpatial.Enabled = False
```

```
        FrmSpatial.framData.Enabled = False
        FrmSpatial.framEdge.Enabled = False
        FrmSpatial.framComp.Enabled = False

        FrmSpatial.LblNumCompt.Enabled = False
        FrmSpatial.TxtNumCompt.Enabled = False
        FrmSpatial.LblScaling.Enabled = False
        FrmSpatial.txtScaling.Enabled = False
        FrmSpatial.LblStdDevNum.Enabled = False
        FrmSpatial.txtStdDevNum.Enabled = False
        FrmSpatial.OptGaussian.Enabled = False
        FrmSpatial.OptGauThesh.Enabled = False
        FrmSpatial.ComboNumCompt.Enabled = False
        FrmSpatial.lblMean.Enabled = False
        FrmSpatial.TxtMean.Enabled = False
        FrmSpatial.LblStd.Enabled = False
        FrmSpatial.TxtStd.Enabled = False

        FrmSpatial.OptLinear.Enabled = False
        FrmSpatial.OptExp.Enabled = False

        FrmSpatial.OptLaplacian.Enabled = False
        FrmSpatial.OptRoberts.Enabled = False
        FrmSpatial.OptSobel.Enabled = False
        FrmSpatial.lblEdgeThres.Enabled = False
        FrmSpatial.txtEdgeThres.Enabled = False

    ElseIf ChkSpatial.Value = 1 Then
        FrmSpatial.FraSpatial.Enabled = True
        FrmSpatial.framData.Enabled = True
        FrmSpatial.framEdge.Enabled = True
        FrmSpatial.framComp.Enabled = True

        FrmSpatial.LblNumCompt.Enabled = True
        FrmSpatial.TxtNumCompt.Enabled = True
        FrmSpatial.LblScaling.Enabled = True
        FrmSpatial.txtScaling.Enabled = True
        FrmSpatial.LblStdDevNum.Enabled = True
        FrmSpatial.txtStdDevNum.Enabled = True
        FrmSpatial.OptGaussian.Enabled = True
        FrmSpatial.OptGauThesh.Enabled = True
        FrmSpatial.ComboNumCompt.Enabled = True
        FrmSpatial.lblMean.Enabled = True
        FrmSpatial.TxtMean.Enabled = True
        FrmSpatial.LblStd.Enabled = True
        FrmSpatial.TxtStd.Enabled = True

        FrmSpatial.OptLinear.Enabled = True
        FrmSpatial.OptExp.Enabled = True

        FrmSpatial.OptRoberts.Enabled = True
        FrmSpatial.OptSobel.Enabled = True
        FrmSpatial.lblEdgeThres.Enabled = True
        FrmSpatial.txtEdgeThres.Enabled = True

    End If

    End Sub
```

ComboCompNums_Click

Qualifiers: Private

```
    Private Sub ComboCompNums_Click()

        Dim V As Integer

        V = Val(ComboCompNums.Text)
        TxtThreshold.Text = Threshold(V)
        OldTracerIndex = Val(ComboCompNums.Text) - 1
```

```
End Sub
```

ComboNumCompt_Click

Qualifiers: Private

```
Private Sub ComboNumCompt_Click()
    Dim V As Integer, I As Integer

    V = Val(ComboNumCompt.Text)
    TxtStd.Text = ComptParameter(1, V - 1)
    TxtMean.Text = ComptParameter(0, V - 1)
    OldTracerIndex = Val(ComboNumCompt.Text) - 1


End Sub
```

Command1_Click

Qualifiers: Private

```
Private Sub Command1_Click()

If ChkSpatial.Value = 1 Then
'Spatial Analysis ---------------------------------------
    '10-30-96
    If OptLaplacian.Value Then
        EdgeOption = 1
    ElseIf OptRoberts.Value Then
        EdgeOption = 2
    ElseIf OptSobel.Value Then
        EdgeOption = 3
    End If

HistoType = Val(OptGaussian.Value)

    Scaling = Val(txtScaling.Text)
    SpatialStdDevNum = Val(txtStdDevNum.Text)
    DataFormOption = OptExp.Value
    SpatialEdgeThres = Val(txtEdgeThres.Text)
    CompNum = Val(TxtNumCompt.Text)
    Call SpatialAnalysis
'----------------------------------------------------------
End If

'Sampling Partial Sample Domain ------------------------
    If ChkSampling.Value = 1 Then
        OriginNV = Val(txtOriginNV.Text)
        NewNV = Val(txtNewNV.Text)
        Scaling = 1
        If NewNV = 0 And NewNV > NV Then
            Beep
            MsgBox "Incorrect Input"
            Exit Sub
        End If
        Call PartialArraySampling
    Else

    End If
'----------------------------------------------------------


'Histogram ----------------------------------------------
    If ChkHisto.Value = 1 Then
        HistoMax = Val(txtMaxIntense.Text)
```

```
        HistoMin = Val(txtMinIntense.Text)
        ScalingFactor = Val(txtScaleFactor.Text)
        HistoClassNum = Val(txtClassNum.Text)
        CenterX = Val(txtX.Text)
        CenterY = Val(txtY.Text)
        Radius = Val(txtRadius.Text)
        If HistoMin > HistoMax Then
            Beep
            MsgBox "Incorrect Input"
            Exit Sub
        End If
        Call GenHistogram
    End If
'-----------------------------------------------------------

'Direct Thresholding Processing --------------------------
    If ChkDirect.Value = 1 Then
     Scaling = Val(TxtScales.Text)
        Call DirectThresholdingAnalysis
    End If
'-----------------------------------------------------------
If ChkVariogram.Value = 1 Then
    Call VariogramSemivariance
End If

If ChkLowPass.Value = 1 Then
    Call LowPass
End If

End Sub
```

Image1_Click

Qualifiers: Private

```
Private Sub Image1_Click()

End Sub
```

mnuExit_Click

Qualifiers: Private

```
Private Sub mnuExit_Click()
    Unload FrmSpatial
    FrmSpatial.Hide
    frmRadon.Show

End Sub
```

txtCompNums_LostFocus

Qualifiers: Private

```
Private Sub txtCompNums_LostFocus()

    Dim V As Integer, I As Integer, Prev As Integer
    Dim Entry

    Prev = CompNums
    V = Val(txtCompNums.Text)
```

```
    If V < 2 Then
        MsgBox "Number of Components should be > 1 !"
        Exit Sub
    End If

    ReDim Threshold(1 To V - 1) As Single
    CompNums = V
    If V > Prev Then 'add items into combo box
            For I = Prev To V - 1
                ComboCompNums.AddItem I
                Threshold(I) = 0.005
            Next I
    ElseIf V <> Prev Then ' ie. V<Prev, remove items from combo box
            For I = V To Prev - 1
                ComboCopmNums.RemoveItem I
            Next I
    End If

End Sub
```

TxtDoloMean_GotFocus

Qualifiers: Private

```
Private Sub TxtDoloMean_GotFocus()
    txtDoloMean.SelStart = 0
    txtDoloMean.SelLength = 65000

End Sub
```

txtEdgeThres_GotFocus

Qualifiers: Private

```
Private Sub txtEdgeThres_GotFocus()
 txtEdgeThres.SelStart = 0
    txtEdgeThres.SelLength = 65000
End Sub
```

TxtGypMean_GotFocus

Qualifiers: Private

```
Private Sub TxtGypMean_GotFocus()
    txtGypMean.SelStart = 0
    txtGypMean.SelLength = 65000

End Sub
```

txtInputFile_Change

Qualifiers: Private

```
Private Sub txtInputFile_Change()
    txtInputFile.SelStart = 0
```

```
    txtInputFile.SelLength = 65000
End Sub
```

txtInputFile_DblClick

Qualifiers: Private

```
Private Sub txtInputFile_DblClick()
    Dim temp$

    frmGetFile.Caption = "Input File Name"
    frmGetFile.FileTypes.AddItem "All Files (*.*)"
    frmGetFile.Show MODAL
    temp$ = frmGetFile.FullPath.Text
    If temp$ <> "" Then
        txtInputFile.Text = UCase$(frmGetFile.Text1)
        'We set a global variable here
        InputFile = UCase$(frmGetFile.FullPath.Text)
    End If


End Sub
```

TxtMean_Change

Qualifiers: Private

```
Private Sub TxtMean_Change()

    V = Val(ComboNumCompt.Text)
    ComptParameter(0, V - 1) = Val(TxtMean.Text)

End Sub
```

TxtMean_GotFocus

Qualifiers: Private

```
Private Sub TxtMean_GotFocus()
    TxtMean.SelStart = 0
    TxtMean.SelLength = 65000
End Sub
```

TxtNumCompt_LostFocus

Qualifiers: Private

```
Private Sub TxtNumCompt_LostFocus()

    Dim V As Integer, I As Integer, Prev As Integer
    Dim Entry

    Prev = NumberOfComponents
    V = Val(TxtNumCompt.Text)
```

```
    If V < 1 Then
        MsgBox "The valid Number of Tracers should greater than 0",
MB_ICONEXCLAMATION
        Exit Sub
    End If

    NumberOfComponents = V
    ReDim ComptParameter(2, V)

    If V > Prev Then 'add items into combo box
            For I = Prev To V - 1
                ComboNumCompt.AddItem Str$(I + 1), I
                ComptParameter(0, I) = 0.015
                ComptParameter(1, I) = 0.0005
            Next I
    ElseIf V <> Prev Then ' ie. V<Prev, remove items from combo box
            For I = V To Prev - 1
                ComboNumCompt.RemoveItem I
            Next I
    End If

End Sub
```

TxtScale_GotFocus

Qualifiers: Private

```
Private Sub TxtScale_GotFocus()
    TxtScale.SelStart = 0
    TxtScale.SelLength = 65000
End Sub
```

TxtStd_Change

Qualifiers: Private

```
Private Sub TxtStd_Change()

    V = Val(ComboNumCompt.Text)
     ComptParameter(1, V - 1) = Val(TxtStd.Text)

End Sub
```

TxtStd_GotFocus

Qualifiers: Private

```
Private Sub TxtStd_GotFocus()
    TxtStd.SelStart = 0
    TxtStd.SelLength = 65000
End Sub
```

txtStdDev_Change

Qualifiers: Private

```
Private Sub txtStdDev_Change()
End Sub
```

TxtStdDevNum_GotFocus

Qualifiers: Private

```
Private Sub TxtStdDevNum_GotFocus()
    txtStdDevNum.SelStart = 0
    txtStdDevNum.SelLength = 65000

End Sub
```

TxtThreshold_Change

Qualifiers: Private

```
Private Sub TxtThreshold_Change()
    V = Val(ComboCompNums.Text)
    Threshold(V) = Val(TxtThreshold.Text)

End Sub
```

TxtThreshold_GotFocus

Qualifiers: Private

```
Private Sub TxtThreshold_GotFocus()
 TxtThreshold.SelStart = 0
    TxtThreshold.SelLength = 65000
End Sub
```

CALC.BAS

Mod Date Tue Sep 02 13:59:29 1997
Size 82362

Declarations

```
Attribute VB_Name = "CALC"
Option Explicit
'*******************************************
' *******  VARIABLE DECLARATIONS  **********
'*******************************************
' A            = image distance between rays
' Conv(K)      = the convoluted projection array.
' ConvI(KR)    = convolution pf counts
' Intens(K)    = count data
' Iy           = Y direction counter for the pixel location in an image array.
' Ix           = X direction counter for the pixel location in an image array.
' Kpad1        = adds 50% to the size of the projection file.
' Kpad2        = adds 150% to the size of the projection file.
' NV           = number of image voxels (pixels) per side (60 or 120).
' Phi(K)       = the filter function value for a given ray.
' Pin          = the angle between projection in radians.
' PinTau       = the PIN times TAU.
' PinTauSqr    = the square of pintau
' Proj(J,K)    = the projection data collected from lab which read in from a file.
' Projs(K)     = the projection data after padding procedure.
' ProjNum      = projectin number (=61) .
' RayCenter    = the center number of ray.
' RayNum       = ray number per view (=61).
' RsyNum2      = doubles the size of the projection file.
' Tau          = distance between rays(mm).
' X(I)         = the X Cartesian coordinate of the voxel.
' Y(Y)         = the Y Cartesian coordinate of the voxel.
' Mu(IX,IY)    = the image data matrix (NV x NV).

Dim Proj() As Single, Intens0() As Long
Dim Conv() As Single, Projs() As Single
Dim Intens() As Single, ConvI() As Single
Dim Phi() As Single, PhiSqr() As Single
Global Mu() As Single, Sigma() As Single
Dim X() As Single, Y() As Single
Dim Ix As Integer, Iy As Integer, L As Integer
Dim RayNum2 As Integer
Dim Kpad1 As Integer, Kpad2 As Integer
Dim R As Single, Pi As Single
Dim RayCenter As Single, A As Single, Pin As Single
Dim temp As Single, TempI As Single
Dim PinTau As Single, Inten0 As Single
Dim PinTauSqr As Single
'**** Others ****

Dim ThetaJ As Single, SinThetaJ As Single
Dim CosThetaJ As Single
'**************************
'**** Global Variables ****
'**************************
'**** Global Input Variables ****

Global InputFile As String
Global Tau As Single
Global NV As Integer
Global ProjNum As Integer
Global RayNum As Integer
Global ShiftError As Single
'Global ImgMax As Single
'Global ImgMin As Single
```

```
'4-12-96 hsieh

Global NewNV As Integer
Global OriginNV As Integer
'7-4-96 add Atten. Coeff. Correction

Global AttenCoeff, OutDiameter, InDiameter As Single
Dim NewProj() As Single
Dim NewIntens0() As Single
Dim NewIntens() As Single
Dim CorrIntens() As Single
Dim Xdist() As Single
'8-19-96 hsieh
'add histogram procedure into the rad-2-4 program

    Global HistoMax    As Single
    Global HistoMin    As Single
    Global ScalingFactor As Single
    Global HistoBinSize  As Single
    Global HistoClassNum As Integer
    Global SpatialDoloMean As Single
    Global SpatialGypMean  As Single
    Global SpatialStdDev As Single
    Global SpatialStdDevNum As Integer
    Global SpatialEdgeThres As Single
    Global Scaling As Single
    Global CenterX As Integer
    Global CenterY As Integer
    Global Radius As Single
    Dim AccumBinSize() As Single
    '10-30-96

    Global DataFormOption As Boolean
    Global EdgeOption As Integer
'9-12-96
'use to identify the voxel belong after spatial
'identificaiton process
' it is assign to 1 if it has been identified, or to 0 for
' a new voxel

    Global VoxelId()  As Integer
'9-20-96 hsieh
' add a "direct thresholding procedure
'select  a threshold and generate a output file.
' all the data points above the threshold are set to A number
' and keep all the others intact.

    Global ThresholdValue1 As Single
    Global ThresholdValue2 As Single
    Global MeanDolomite As Single
    Global MeanGypsum As Single
    Global IndexGypsum, IndexDolomite, IndexVoid As Integer
    Global IndexGypsumSum, IndexDolomiteSum, IndexVoidSum As Single
'9-24-96 hsieh
'Sobel Edge detection algorithm

    Global RadArr() As Single
    Global EdgeArr() As Single
    Global NewEdgeArr() As Single
    Global NewRadArr() As Single
    Global NewNonEdgeRadArr() As Single
    Global NewEdgeRadArr() As Single
    Global NewVoxelArr() As Single
    Global LaplaEdgeArr() As Single
' sub Spatial2  10-13-96 hsieh
'Procedure
    'Define 3x3 subRadArrays that are sampled from part of the whole sample image.
    'The result arrays are used to generate frequency histograms for statistical
analysis
        'of all the components in the image.
'Objective:
    'Separate different attribute voxels (edge or non-dege)
    'Allocate the properties of all the voxels
    'Examine the spatial correlation of the voxels

    Global PartRadArr() As Single
```

```
    Global Const NumOfPartArrayPerSide = 3
'1-30-97 hsieh

Global ComptParameter() As Single
Global UpperBound() As Single
Global LowerBound() As Single
Global NumberOfComponents As Integer
Global CompNum As Integer
Global CompNums As Integer 'for direct thresholding processing 2-5-97
Global Threshold() As Single
Global ScaleDirect As Single
Global HistoType As Boolean
```

Subroutines

CorrectattenCoeff

Qualifiers:	Public		
Arguments:	InputFile	Variant	By Ref.

```
Sub CorrectattenCoeff(InputFile)

    Dim I, j, k, d As Integer
    Dim StrLen
    Dim AirCount1 As Long, AirCount2 As Long
    Dim InputFileNum, FileLength, NextLine

    ReDim Xdist(1 To RayNum) As Single

    ReDim NewProj(1 To ProjNum, 1 To RayNum)
    ReDim NewIntens0(1 To ProjNum)
    ReDim NewIntens(1 To ProjNum, 1 To RayNum)
    ReDim CorrIntens(1 To ProjNum, 1 To RayNum)

'   ***** Redimension the global variables *********
    ReDim Proj(1 To ProjNum, 1 To RayNum) As Single
    ReDim Intens0(1 To ProjNum) As Long
'   ************************************************


    FileLength = ProjNum * (RayNum + 2)
    InputFileNum = FreeFile
    Open InputFile For Input As InputFileNum Len = FileLength

    For j = 1 To ProjNum
        DoEvents
        Line Input #InputFileNum, NextLine
        AirCount1 = Val(NextLine)
        Line Input #InputFileNum, NextLine
        AirCount2 = Val(NextLine)
        Intens0(j) = (AirCount1 + AirCount2) / 2
        For k = 1 To RayNum
            Line Input #InputFileNum, NextLine
            Proj(j, k) = Val(NextLine)
        Next k
    Next j
    Close #InputFileNum

    For I = 1 To ProjNum
        For j = 1 To RayNum
            NewIntens(I, j) = Intens0(I) / Exp(Proj(I, j))
        Next j
    Next I

    For j = 1 To RayNum
        d = Abs((RayNum - 1) * Tau / 2 + Tau * ShiftError - Tau * (j - 1))
        'If j <= (RayNum / 2) Then
            'd = 22.25 - 0.5*j + 0.55
        '    d = .55 + (tau * (RayNum - 1) / 2) - tau * j
```

```
        'Else
            'd = 0.55 + 0.5 * (j - 45)
        '    d = .55 + tau * (j - RayNum / 2)
        'End If
        Xdist(j) = 2 * (Sqr((OutDiameter / 2) ^ 2 - (d) ^ 2) - Sqr((InDiameter / 2) ^
2 - (d) ^ 2))
    Next j

    For I = 1 To ProjNum
        For j = 1 To RayNum
            CorrIntens(I, j) = NewIntens(I, j) / Exp(-Xdist(j) * AttenCoeff)
        Next j
        NewIntens0(I) = (CorrIntens(I, 1) + CorrIntens(I, RayNum)) / 2
    Next I

    Dim FileNum, OutputFile
    StrLen = Len(InputFile)
    OutputFile = Mid$(InputFile, 1, StrLen - 4)
    OutputFile = OutputFile & "N" & ".DAT"
    FileNum = FreeFile
    Open OutputFile For Output As #FileNum

    For I = 1 To ProjNum
        Print #FileNum, CorrIntens(I, 1)
        Print #FileNum, CorrIntens(I, RayNum)
        For j = 1 To RayNum
            NewProj(I, j) = Log(NewIntens0(I) / CorrIntens(I, j))
            Print #FileNum, NewProj(I, j)
        Next j
    Next I
    Close #FileNum

    MsgBox "Correction Completed!"

End Sub
```

DirectThresholdingAnalysis

Qualifiers: Public

```
Sub DirectThresholdingAnalysis()
    Dim I, j, k As Integer
    Dim ArraySize As Integer

    Dim RadArr() As Single
    'Dim NewRadArr() As Single

    Dim VolFract() As Integer
    ReDim VolFract(1 To CompNums) As Integer

    Call ReadRadInput(ArraySize, RadArr())

    ReDim VoxelId(1 To ArraySize, 1 To ArraySize) As Integer
    ReDim NewRadArr(1 To ArraySize, 1 To ArraySize) As Single
    Dim lower, upper, upperest As Single

    DoEvents

    lower = -100#
    upperest = 100#
        For I = 1 To CompNums
            VolFract(I) = 0
        Next I
    Dim IndexArr() As Integer
    ReDim IndexArr(1 To ArraySize, 1 To ArraySize) As Integer

    For j = 1 To ArraySize
            For k = 1 To ArraySize
                IndexArr(k, j) = 0
        Next k
```

```
    Next j
'9-2-97=============================
    For I = 1 To CompNums
        If I <> CompNums Then
            upper = Threshold(I)
        Else
            upper = upperest
        End If

        For j = 1 To ArraySize
            For k = 1 To ArraySize
                If lower <= RadArr(k, j) And RadArr(k, j) < upper And IndexArr(k, j)
= 0 Then
                    NewRadArr(k, j) = I * 1#                    '2-5-97    hsieh
                    IndexArr(k, j) = 1
                    VolFract(I) = VolFract(I) + 1
                End If
            Next k
        Next j
        lower = upper
    Next I
'9-2-97=============================

    Dim IndexDolomiteSum, IndexGypsumSum, IndexVoidSum As Single

    IndexDolomiteSum = VolFract(1)
    IndexGypsumSum = VolFract(2)
    IndexVoidSum = 1 - IndexDolomiteSum - IndexGypsumSum

    Call WriteOutputSingle(InputFile, ".DIR", ArraySize, NewRadArr())

    MsgBox "Procedure Completed!"
End Sub
```

DoCalculations

Qualifiers: Public

```
Sub DoCalculations()
    Dim I As Integer, j As Integer
    Dim k As Integer, KR As Integer, Kabs As Integer
    Dim InputFileNum, FileLength, NextLine
    Dim KD As Single
    Dim XSinThetaJ() As Single, YCosThetaJ() As Single
    Dim L1R As Single, RL As Single
    Dim XPos, YPos As Single

    '**** Calculate Constants ****

    Pi = 3.14159265
    RayNum2 = 2 * RayNum - 1
    RayCenter = (RayNum2 / 2#) + 0.5 + ShiftError
    Kpad1 = Int(RayNum / 2)
    Kpad2 = RayNum + Kpad1

    XPos = Tau / 2
    YPos = XPos

'    A = 2# / (RayNum2 - 1)      original
    A = Tau * 2 / (RayNum2 - 1)
    Pin = Pi / (ProjNum)
    PinTau = Pin * Tau
    PinTauSqr = PinTau * PinTau


    '**** Redimension all arrays ****

    ReDim Proj(1 To ProjNum, 1 To RayNum)
    ReDim Intens0(1 To ProjNum)
    ReDim Conv(1 To (RayNum2 + 2))
    ReDim Projs(1 To (RayNum2 + 2))
```

```
ReDim Intens(1 To (RayNum2 + 2))
ReDim ConvI(1 To (RayNum2 + 2))

ReDim Phi(0 To RayNum2)
ReDim PhiSqr(0 To RayNum2)
ReDim X(1 To NV)
ReDim Y(1 To NV)
ReDim XSinThetaJ(1 To NV)
ReDim YCosThetaJ(1 To NV)
ReDim Mu(1 To NV, 1 To NV)
ReDim Sigma(1 To NV, 1 To NV)

'**** Calculate X(I), Y(I) - X and Y Coordinate of Voxel ****
For I = 1 To NV
    X(I) = -XPos + (I - XPos) / NV
    Y(I) = -YPos + (I - YPos) / NV
Next I

'**** Filter Function (Kak & Slaney, 1988) ****

Phi(0) = 1 / (4# * Tau * Tau)
PhiSqr(0) = Phi(0) * Phi(0)

For k = 2 To RayNum2 Step 2
    Phi(k) = 0#
    PhiSqr(k) = 0#
Next k

temp = -1 / (Pi * Pi * Tau * Tau)
For k = 1 To RayNum2 Step 2
    KD = k
    Phi(k) = temp / (KD * KD)
    PhiSqr(k) = Phi(k) * Phi(k)
Next k

'**** Read Data File ****

Call ReadInputFile(InputFile)

'**** Initialize Mu(IX, IY) & Sigma(IX, IY) ****

For Iy = 1 To NV
    For Ix = 1 To NV
        Mu(Ix, Iy) = 0#
        Sigma(Ix, Iy) = 0#
    Next Ix
Next Iy

'**** ********************** ****

For j = 1 To ProjNum
    frmRadon.lblPercentage.Caption = Format$(CInt(j * 100 / ProjNum)) & "%"
    DoEvents
    ThetaJ = (j - 1) * Pin
    CosThetaJ = Cos(ThetaJ)
    SinThetaJ = Sin(ThetaJ)
    Inten0 = Intens0(j)

'**** Pad Projection With Zero Rays ****

    For k = 1 To Kpad1
        Projs(k) = 0
        Intens(k) = Inten0
    Next k

    For k = (Kpad1 + 1) To Kpad2
        Projs(k) = Proj(j, k - Kpad1)
        Intens(k) = Inten0 * Exp(-Projs(k))
    Next k

    For k = (Kpad2 + 1) To RayNum2
        Projs(k) = 0
        Intens(k) = Inten0
    Next k

    '**** Convolution (Hilbert Transform) ****
```

```
        For KR = 1 To RayNum2
            temp = 0#
            TempI = 0#
            For k = 1 To RayNum2
                Kabs = Abs(KR - k) 'Iabs Changed to Abs
                temp = temp + (Projs(k) * Phi(Kabs))
                TempI = TempI + (PhiSqr(Kabs) / Intens(k))
            Next k
            Conv(KR) = temp
            ConvI(KR) = TempI
        Next KR

        '**** Back Projection ****
        For I = 1 To NV
            YCosThetaJ(I) = Y(I) * CosThetaJ
            XSinThetaJ(I) = X(I) * SinThetaJ
        Next I

        For Iy = 1 To NV
            For Ix = 1 To NV
                R = RayCenter + (YCosThetaJ(Iy) + XSinThetaJ(Ix)) / A
                L = Int(R)  'R changes to INT(R)
                L1R = L + 1 - R
                RL = R - L
'========7-29-97 added by hsieh
                If (L <= 0 Or L >= RayNum2 + 2) Then
                    GoTo Jump2Ix
                End If
'========7-29-97 added by hsieh
                Mu(Ix, Iy) = Mu(Ix, Iy) + L1R * Conv(L) + RL * Conv(L + 1)
                Sigma(Ix, Iy) = Sigma(Ix, Iy) + L1R * ConvI(L) + RL * ConvI(L + 1)
Jump2Ix:
            Next Ix
        Next Iy
    Next j

    '**** Mulitiply Mu And Sigma by Pintau ***

    For Iy = 1 To NV
        For Ix = 1 To NV
            Mu(Ix, Iy) = PinTau * Mu(Ix, Iy)
            Sigma(Ix, Iy) = PinTauSqr * Sigma(Ix, Iy)
        Next Ix
    Next Iy

End Sub
```

EdgeDetection

Qualifiers: Public

```
Sub EdgeDetection()
'Roberts edge detection algorithm
'see paper 2 for details

    Dim j, k, L As Integer
    Dim ArraySize As Integer

    Dim InputFileNum, OutFileNum, FileLength, NextLine
    Dim StrLen, OutputFile
    Dim FileNum

'9-24-96 hsieh -----------------------------------
'Read in original .rad file to RadArr()
'file starts with the ArraySize, then ArraySize x ArraySize voxels

    InputFileNum = FreeFile
    Open InputFile For Input As InputFileNum 'Len = FileLength

    DoEvents
```

```
    Line Input #InputFileNum, NextLine
    ArraySize = Val(NextLine)

    ReDim RadArr(1 To ArraySize, 1 To ArraySize) As Single
    ReDim NewRadArr(1 To ArraySize, 1 To ArraySize) As Single

    ReDim EdgeArr(1 To ArraySize + 1, 1 To ArraySize + 1) As Single
    ReDim NewEdgeArr(1 To ArraySize + 1, 1 To ArraySize + 1) As Single

    ReDim NewEdgeRadArr(1 To ArraySize + 1, 1 To ArraySize + 1) As Single
    ReDim NewNonEdgeRadArr(1 To ArraySize + 1, 1 To ArraySize + 1) As Single

    ReDim LaplaEdgeArr(1 To ArraySize, 1 To ArraySize) As Single

'read original rad file
    For j = 1 To ArraySize
        For k = 1 To ArraySize
            Line Input #InputFileNum, NextLine
            RadArr(k, j) = Val(NextLine)
        Next k
    Next j
    Close #InputFileNum

'9-24-96 hsieh ---------------------------------------
' Edge Detection - Roberts Operator
' Generate EdgeArr()
    'Call Roberts(ArraySize)

'9-27-96 hsieh ---------------------------------------
' Edge Detection - Sobel Operator
' Generate EdgeArr()
 '   Call Sobel(ArraySize)

'9-24-96 hsieh
'set edge thresholding =0.015
'generate a newedge array defines edge and non-edge voxels
'for ---- Sobel ----- & ----- Roberts -----------

    For j = 1 To ArraySize
        For k = 1 To ArraySize
            If EdgeArr(j, k) > 0.015 Then
                NewEdgeArr(j, k) = 1#
            Else
                NewEdgeArr(j, k) = 0#
            End If
        Next k
    Next j

'10-3-96 hsieh ---------------------------------------
' Edge Detection - Laplacian Operator
' Generate EdgeArr()
'   Call Laplacian(ArraySize)

'   Call WriteOutput(InputFile, "A.LAP", ArraySize, NewRadArr())

'9-24-96 hsieh
'10-4-96 hsieh
'edge voxel     ==> EdgeArr()=1
'non-edge voxel ==> EdgeArr()=0
'replace all non-edge voxels to 0 and show ONLY the density of the edge voxel.
    For j = 1 To ArraySize
        For k = 1 To ArraySize
            If NewEdgeArr(j, k) = 1# Then
                NewEdgeRadArr(j, k) = RadArr(j, k)
            Else ' that is, NewEdgeArr(j, k) = 0#
                NewEdgeRadArr(j, k) = 0#
            End If
        Next k
    Next j

'9-24-96 hsieh
'10-4-96 hsieh
'show ONLY the density of the NON-edge voxels.
'replace all edge voxels to 0.
    For j = 1 To ArraySize
        For k = 1 To ArraySize
```

```
            If NewEdgeArr(j, k) = 0 Then
                NewNonEdgeRadArr(j, k) = RadArr(j, k)
            Else
                NewNonEdgeRadArr(j, k) = 0#
            End If
        Next k
    Next j

'9-24-96 hsieh ------------------------------------
'write edge image to the file - EdgeArr()
'set all non-edge voxels to 0

    Call WriteOutputSingle(InputFile, ".EDG", ArraySize, EdgeArr())

'9-24-96 hsieh ------------------------------------
'9-26-96 hsieh ------------------------------------
'write voxels at edge file to the file - NewEdgeRadArr()
'write the non-edge voxels to the file - NewNonEdgeRadArr()

'set all non-edge voxels to 0

   Call WriteOutputSingle(InputFile, "V.VAE", ArraySize, NewEdgeRadArr())

'set all edge voxels to 0

    Call WriteOutputSingle(InputFile, "N.VAE", ArraySize, NewNonEdgeRadArr())

'9-26-96 hsieh ------------------------------------

    Call EdgeVoxelCDP(ArraySize)
    Call WriteOutputSingle(InputFile, ".VAD", ArraySize, NewVoxelArr())


'9-26-96 hsieh ------------------------------------
'write new non-edge voxel file - NewRadArr
'set all edge voxels to 0

'    Call WriteOutput(InputFile, ".lap", ArraySize, LaplaEdgeArr())

'--------------------------------------------------

End Sub
```

EdgeVoxelCDP

Qualifiers:	Public		
Arguments:	Size	Integer	By Ref.

```
Sub EdgeVoxelCDP(Size As Integer)

    Dim j, k As Integer
    ReDim NewVoxelArr(1 To Size, 1 To Size) As Single

    For j = 1 To Size
        For k = 1 To Size
            If ((j <> 1) And (k <> 1)) And ((j <> Size) And (k <> Size)) Then
            '    If (NewEdgeRadArr(k, j) > (NewEdgeRadArr(k + 1, j)) / 2) Then
            '        NewVoxelArr(k, j) = Max(NewEdgeRadArr(k + 1, j), NewEdgeRadArr(k
- 1, j))
            '    Else
            '        NewVoxelArr(k, j) = Min(NewEdgeRadArr(k + 1, j), NewEdgeRadArr(k
- 1, j))
            '    End If
            'ElseIf j = ArraySize Or k = ArraySize Then
                If (NewEdgeRadArr(k, j) > (NewEdgeRadArr(k + 1, j) + NewEdgeRadArr(k
- 1, j)) / 2) Then
                    NewVoxelArr(k, j) = max(NewEdgeRadArr(k + 1, j), NewEdgeRadArr(k
- 1, j))
                Else
```

```
                    NewVoxelArr(k, j) = Min(NewEdgeRadArr(k + 1, j), NewEdgeRadArr(k
- 1, j))
                End If
            Else
                'If (NewEdgeRadArr(k, j) > (NewEdgeRadArr(k + 1, j) + NewEdgeRadArr(k
- 1, j)) / 2) Then
                    NewVoxelArr(k, j) = NewEdgeRadArr(k, j)
                'Else
                '    NewVoxelArr(k, j) = Min(NewEdgeRadArr(k + 1, j), NewEdgeRadArr(k
- 1, j))
                'End If
            End If
        Next k
    Next j

End Sub
```

GenHistogram

Qualifiers: Public

```
Sub GenHistogram()
'    Global HistoMax     As Single
'    Global HistoMin     As Single
'    Global HistoBinSize  As Single
'    Global HistoClassNum As Integer
'    Global HistoDoloMean As Single
'    Global HistoGypMean  As Single
    Dim I, j, k As Integer
    Dim ArraySize As Integer

    Dim HalfLength As Integer
    Dim InputFileNum, OutFileNum, FileLength, NextLine
    Dim StrLen, OutputFile
    Dim FileNum
    Dim RadArr() As Single

    HistoMax = ScalingFactor * HistoMax
    HistoMin = ScalingFactor * HistoMin
    HistoBinSize = (HistoMax - HistoMin) / (HistoClassNum - 1#)

    'FileLength = OriginNV * OriginNV + 1
    InputFileNum = FreeFile
    Open InputFile For Input As InputFileNum 'Len = FileLength

    DoEvents

    Line Input #InputFileNum, NextLine
    ArraySize = Val(NextLine)
    ReDim RadArr(1 To ArraySize, 1 To ArraySize) As Single

    For j = 1 To ArraySize
        For k = 1 To ArraySize
            Line Input #InputFileNum, NextLine
            RadArr(k, j) = Val(NextLine) * ScalingFactor
        Next k
    Next j
    Close #InputFileNum

    ReDim AccumBinSize(1 To HistoClassNum + 2) As Single

    Dim sumcount, count() As Integer
    ReDim count(1 To HistoClassNum + 2) As Integer

    For I = 1 To HistoClassNum + 2
        AccumBinSize(I) = HistoBinSize * (I - 1#) + HistoMin
        count(I) = 0
    Next I

'12-19-96 hsieh
'define a procedure that samples a circular data point within a certain radius
```

```
    'If CenterX > Int(ArraySize / 2) Or CenterY > Int(ArraySize / 2) Then
    '    CenterX = Int(ArraySize / 2)
    '    CenterY = Int(ArraySize / 2)
    '    Radius = 0.8 * ArraySize / 2
    ' End If

    Dim SelectedVoxel() As Integer
    ReDim SelectedVoxel(1 To ArraySize, 1 To ArraySize) As Integer

    For j = 1 To ArraySize
        For k = 1 To ArraySize
            If Radius >= Sqr((k - CenterX) * (k - CenterX) + (j - CenterY) * (j -
CenterY)) Then
                SelectedVoxel(k, j) = 1
            Else
                SelectedVoxel(k, j) = 0
                RadArr(k, j) = -10#
            End If
        Next k
    Next j

    For j = 1 To ArraySize
        For k = 1 To ArraySize
            If SelectedVoxel(k, j) = 1 Then
                If RadArr(k, j) <= HistoBinSize * HistoClassNum Then
                    For I = 1 To HistoClassNum + 1
                        If RadArr(k, j) <= AccumBinSize(I) Then
                            count(I) = count(I) + 1
                            GoTo NextVoxel
                        End If
                    Next I
                Else
                    count(HistoClassNum + 2) = count(HistoClassNum + 2) + 1
                End If
NextVoxel:
            End If
        Next k
    Next j

    sumcount = 0
    For I = 1 To HistoClassNum + 2
        sumcount = sumcount + count(I)
    Next I

    StrLen = Len(InputFile)
    OutputFile = Mid$(InputFile, 1, StrLen - 4)

    OutFileNum = FreeFile
    OutputFile = OutputFile & ".his"
    Open OutputFile For Output As #OutFileNum

    Print #OutFileNum, "AccumBinSize", "count"
    For I = 1 To HistoClassNum + 2
        Print #OutFileNum, AccumBinSize(I), count(I)
    Next I
    Close #OutFileNum
'   Test selected voxel Image - C()
    Call WriteOutputSingle(InputFile, ".RAH", ArraySize, RadArr())

End Sub
```

Laplacian

Qualifiers:	Public		
Arguments:	InputArr	Single	By Ref.
	Size	Integer	By Value
	EdgeOut	Single	By Ref.

```
Sub Laplacian(InputArr() As Single, ByVal Size As Integer, EdgeOut() As Single)
```

```
    Dim j, k As Integer
    Dim Gx, Gy As Single
    Dim NewS, NewE As Integer
    NewS = 0
    NewE = Size + 1

    For j = NewS To NewE
        For k = NewS To NewE
            If (j = NewS And k = NewS) Then
                    EdgeOut(k, j) = 2# * RadArr(k, j) - 1# * (RadArr(k + 1, j) +
RadArr(k, j + 1))
            ElseIf (j = NewS And k = NewE) Then
                    EdgeOut(k, j) = 2# * RadArr(k, j) - 1# * (RadArr(k - 1, j) +
RadArr(k, j + 1))
            ElseIf (j = NewE And k = NewS) Then
                    EdgeOut(k, j) = 2# * RadArr(k, j) - 1# * (RadArr(k + 1, j) +
RadArr(k, j - 1))
            ElseIf (j = NewE And k = NewE) Then
                    EdgeOut(k, j) = 2# * RadArr(k, j) - 1# * (RadArr(k - 1, j) +
RadArr(k, j - 1))
            ElseIf (j = NewS And k <> NewS And k <> NewE) Then
                    EdgeOut(k, j) = 3# * RadArr(k, j) - 1# * (RadArr(k + 1, j) +
RadArr(k - 1, j) + RadArr(k, j + 1))
            ElseIf (j = NewE And k <> NewS And k <> NewE) Then
                    EdgeOut(k, j) = 3# * RadArr(k, j) - 1# * (RadArr(k + 1, j) +
RadArr(k - 1, j) + RadArr(k, j - 1))
            ElseIf (k = NewS And j <> NewE And j <> NewS) Then
                    EdgeOut(k, j) = 3# * RadArr(k, j) - 1# * (RadArr(k + 1, j) +
RadArr(k, j + 1) + RadArr(k, j - 1))
            ElseIf (k = NewE And j <> NewE And k <> NewS) Then
                    EdgeOut(k, j) = 3# * RadArr(k, j) - 1# * (RadArr(k - 1, j) +
RadArr(k, j + 1) + RadArr(k, j - 1))
            Else
                    EdgeOut(k, j) = 4# * RadArr(k, j) - 1# * (RadArr(k + 1, j) +
RadArr(k - 1, j) + RadArr(k, j + 1) + RadArr(k, j - 1))
            End If
        Next k
    Next j

End Sub
```

LowPass

Qualifiers: Public

```
Sub LowPass()
    Dim I, j, k As Integer
    Dim ArraySize As Integer
    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutputFile
    Dim OriginArr()
    Dim OriginSingle() As Single
    Dim OriginalStart, OriginalEnd, NewStart, NewEnd As Integer

    Call ReadInputInteger(ArraySize, OriginArr())

    ReDim OriginSingle(1 To ArraySize, 1 To ArraySize)

    For j = 1 To ArraySize
        For k = 1 To ArraySize
            OriginSingle(j, k) = OriginArr(j, k) * 1#
        Next k
    Next j

    Call PaddingRadArr(ArraySize, OriginSingle(), RadArr())

    OriginalStart = 0
    OriginalEnd = ArraySize + 1
    NewStart = 1
    NewEnd = ArraySize
```

```
    Dim position() As Single
    ReDim position(1 To 8) As Single

    Dim SmoothVoxel() As Integer
    ReDim SmoothVoxel(NewStart To NewEnd, NewStart To NewEnd) As Integer

    Dim SmoothVoxelSingle() As Single
    ReDim SmoothVoxelSingle(NewStart To NewEnd, NewStart To NewEnd) As Single

    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
                    position(1) = RadArr(k - 1, j - 1)
                    position(2) = RadArr(k, j - 1)
                    position(3) = RadArr(k + 1, j - 1)
                    position(4) = RadArr(k - 1, j)
                    position(5) = RadArr(k + 1, j)
                    position(6) = RadArr(k - 1, j + 1)
                    position(7) = RadArr(k, j + 1)
                    position(8) = RadArr(k + 1, j + 1)
                    SmoothVoxel(k, j) = Int((position(1) + position(2) + position(3)
+ position(4) + position(5) + position(6) + position(7) + position(8)) / 8)
                    SmoothVoxelSingle(k, j) = (position(1) + position(2) +
position(3) + position(4) + position(5) + position(6) + position(7) + position(8)) /
8
        Next k
    Next j
'   1. Component Image - C()
    Call WriteOutputInteger(InputFile, "I.LOW", ArraySize, SmoothVoxel())
    Call WriteOutputSingle(InputFile, "S.LOW", ArraySize, SmoothVoxelSingle())
    MsgBox "Procedure Completed!"
End Sub
```

PaddingRadArr

Qualifiers:	Public		
Arguments:	Size	Integer	By Value
	OriginRadArr	Single	By Ref.
	InputArr	Single	By Ref.

```
Sub PaddingRadArr(ByVal Size As Integer, OriginRadArr() As Single, InputArr() As
Single)

    Dim k, j As Integer
    ReDim InputArr(0 To Size + 1, 0 To Size + 1) As Single

    For j = 0 To Size + 1
        For k = 0 To Size + 1
            InputArr(k, j) = 0#
        Next k
    Next j

    For j = 1 To Size
        For k = 1 To Size
            InputArr(k, j) = OriginRadArr(k, j)
        Next k
    Next j

    InputArr(0, 0) = OriginRadArr(1, 1)
    InputArr(Size + 1, Size + 1) = OriginRadArr(Size, Size)
    InputArr(0, Size + 1) = OriginRadArr(1, Size)
    InputArr(Size + 1, 0) = OriginRadArr(Size, 1)

    For k = 1 To Size
        InputArr(k, 0) = OriginRadArr(k, 1)
        InputArr(k, Size + 1) = OriginRadArr(k, Size)
    Next k

    For j = 1 To Size
        InputArr(0, j) = OriginRadArr(1, j)
```

```
        InputArr(Size + 1, j) = OriginRadArr(Size, j)
    Next j

End Sub
```

PartialArraySampling

Qualifiers: Public

```
Sub PartialArraySampling()

    Dim j As Integer, k As Integer
    Dim NVSize As Integer
    Dim HalfLength As Integer

    Dim RadArr() As Single
    Dim NewRadArr() As Single
    ReDim NewRadArr(1 To NewNV, 1 To NewNV) As Single

    HalfLength = (OriginNV - NewNV) / 2

    Call ReadRadInput(NVSize, RadArr())

    '=======================================================
    ' sample only the desired portion of a image and save to
    ' a new file " *.NEW "
    '=======================================================
    For j = 1 To NVSize
        For k = 1 To NVSize
            If (HalfLength + 1 <= j And j <= HalfLength + NewNV And HalfLength + 1 <=
k And k <= HalfLength + NewNV) Then
                NewRadArr(k - HalfLength, j - HalfLength) = RadArr(k, j)
            End If
        Next k
    Next j

    Call WriteOutputSingle(InputFile, ".NEW", NewNV, NewRadArr())

End Sub
```

ReadInputFile

Qualifiers: Public
Arguments: InputFile Variant By Ref.

```
Sub ReadInputFile(InputFile)
    Dim j, k, jj, kk As Integer
    Dim AirCount1 As Long, AirCount2 As Long
    Dim IntensTemp, Intens0Temp As Long
    Dim InputFileNum, FileLength, NextLine

    FileLength = ProjNum * (RayNum + 2)
    InputFileNum = FreeFile
    Open InputFile For Input As InputFileNum Len = FileLength
    Dim index As Integer

'===============================================
'Original file Read-In processing

    'For j = 1 To ProjNum
    '     DoEvents
    '      Line Input #InputFileNum, NextLine
    '       AirCount1 = Val(NextLine)
    '        Line Input #InputFileNum, NextLine
    '        AirCount2 = Val(NextLine)
    '        Intens0(j) = (AirCount1 + AirCount2) / 2
```

```
'        For k = 1 To RayNum
'        Line Input #InputFileNum, NextLine
'         Proj(j, k) = Val(NextLine)
'      Next k

'==================================================================
'8-8-96 by hsieh
'By changing ProjNum and/or RayNum reading sequence the orientation
'of the reconstructed image can be adjusted.
'
'   ProjNum  |   RayNum    |        Orientation
' reading seq| reading seq|
'----------------------------------------------------------
'   Min->Max |   Min->Max | View beneath/from source side
'   Max->Min |   Min->Max | View Top/from detector side
'   Max->Min |   Max->Min | View Top/from source side
'==================================================================

    For j = ProjNum To 1 Step -1
        DoEvents
        Line Input #InputFileNum, NextLine
        AirCount1 = Val(NextLine)
        Line Input #InputFileNum, NextLine
        AirCount2 = Val(NextLine)
        Intens0(j) = (AirCount1 + AirCount2) / 2
        For k = RayNum To 1 Step -1
            Line Input #InputFileNum, NextLine
            Proj(j, k) = Val(NextLine)
        Next k


'7-7-96 hsieh
'bolts correction for plane#c3av49 and 50 ======
        'index = 0

        'If AirCount1 < 25000 Then
        '    If Rnd < .5 Then
        '        AirCount1 = 26000 + 200
        '    Else
        '        AirCount1 = 26000 - 200
        '    End If
        '    index = 1
        'End If

        'If AirCount2 < 25000 Then
        '    If Rnd < .5 Then
        '        AirCount2 = 26000 + 200
        '    Else
        '        AirCount2 = 26000 - 200
        '    End If
        '    index = 1
        'End If

        'If index = 1 Then
        '    Intens0Temp = Intens0(j)
        '    Intens0(j) = (AirCount1 + AirCount2) / 2
        '    For kk = 1 To RayNum
        '        IntensTemp = Intens0Temp / Exp(Proj(j, kk))
        '        Proj(j, kk) = Log(Intens0(j) / IntensTemp)
        '    Next
        'End If
'=====================================================

    Next j
    Close #InputFileNum

End Sub
```

ReadInputInteger

Qualifiers: Public

Arguments:	Size	Integer	By Ref.
	InputArr	Variant	By Ref.

```
Sub ReadInputInteger(Size As Integer, InputArr())
    Dim InputFileNum, NextLine
    Dim FileNum
    Dim j, k As Integer

    InputFileNum = FreeFile
    Open InputFile For Input As InputFileNum
    Line Input #InputFileNum, NextLine
    Size = Val(NextLine)
    ReDim InputArr(1 To Size, 1 To Size)

    For j = 1 To Size
        For k = 1 To Size
            Line Input #InputFileNum, NextLine
            InputArr(k, j) = Val(NextLine)
        Next k
    Next j

    Close #InputFileNum

End Sub
```

ReadRadInput

Qualifiers:	Public		
Arguments:	Size	Integer	By Ref.
	InputArr	Single	By Ref.

```
Sub ReadRadInput(Size As Integer, InputArr() As Single)
    Dim InputFileNum, NextLine
    Dim FileNum
    Dim j, k As Integer

    InputFileNum = FreeFile
    Open InputFile For Input As InputFileNum
    Line Input #InputFileNum, NextLine
    Size = Val(NextLine)
    ReDim InputArr(1 To Size, 1 To Size) As Single

    For j = 1 To Size
        For k = 1 To Size
            Line Input #InputFileNum, NextLine
            InputArr(k, j) = Val(NextLine) * Scaling
        Next k
    Next j

    Close #InputFileNum

End Sub
```

Roberts

Qualifiers:	Public		
Arguments:	InputArr	Single	By Ref.
	Size	Integer	By Value
	EdgeOut	Single	By Ref.

```
Sub Roberts(InputArr() As Single, ByVal Size As Integer, EdgeOut() As Single)
```

```
    Dim j, k As Integer
    Dim NewS, NewE As Integer

    NewS = 0
    NewE = Size + 1

    For j = NewS To NewE
        For k = NewS To NewE
            If (j <> NewE And k <> NewE) Then
                EdgeOut(k, j) = Abs(InputArr(k, j) - InputArr(k + 1, j + 1)) +
Abs(InputArr(k, j + 1) - InputArr(k + 1, j))
            ElseIf (j = NewE And k <> NewE) Then
                 EdgeOut(k, j) = Abs(InputArr(k, j)) + Abs(InputArr(k + 1, j))
            ElseIf (k = NewE And j <> NewE) Then
                 EdgeOut(k, j) = Abs(InputArr(k, j)) + Abs(InputArr(k, j + 1))
            Else
                EdgeOut(k, j) = InputArr(k, j)
            End If
        Next k
    Next j

End Sub
```

Sobel

Qualifiers:	Public		
Arguments:	InputArr	Single	By Ref.
	Size	Integer	By Value
	EdgeOut	Single	By Ref.

```
Sub Sobel(InputArr() As Single, ByVal Size As Integer, EdgeOut() As Single)

    Dim j, k As Integer
    Dim Gx, Gy As Single
    Dim NewS, NewE As Integer
    NewS = 0
    NewE = Size + 1

    For j = NewS To NewE
        For k = NewS To NewE
            If (j = NewS And k = NewS) Or (j = NewS And k = NewE) Or (j = NewE And k
= NewS) Or (j = NewE And k = NewE) Then
                    EdgeOut(k, j) = InputArr(k, j)
            ElseIf (j = NewS And k <> NewS And k <> NewE) Then
                    Gy = Abs(2 * InputArr(k + 1, j) + InputArr(k + 1, j + 1) - (2 *
InputArr(k - 1, j) + InputArr(k - 1, j + 1)))
                    EdgeOut(k, j) = Gy
            ElseIf (j = NewE And k <> NewS And k <> NewE) Then
                    Gy = Abs(2 * InputArr(k + 1, j) + InputArr(k + 1, j - 1) - (2 *
InputArr(k - 1, j) + InputArr(k - 1, j - 1)))
                    EdgeOut(k, j) = Gy
            ElseIf (k = NewS And j <> NewE And j <> NewS) Then
                    Gx = Abs(2 * InputArr(k, j + 1) + InputArr(k + 1, j + 1) - (2 *
InputArr(k, j - 1) + InputArr(k + 1, j - 1)))
                    EdgeOut(k, j) = Gx
            ElseIf (k = NewE And j <> NewE And k <> NewS) Then
                    Gx = Abs(2 * InputArr(k, j + 1) + InputArr(k - 1, j + 1) - (2 *
InputArr(k, j - 1) + InputArr(k - 1, j - 1)))
                    EdgeOut(k, j) = Gx
            Else
                    Gx = Abs(InputArr(k + 1, j - 1) + 2 * InputArr(k + 1, j) +
InputArr(k + 1, j + 1) - (InputArr(k - 1, j - 1) + 2 * InputArr(k - 1, j) +
InputArr(k - 1, j + 1)))
                    Gy = Abs(InputArr(k - 1, j + 1) + 2 * InputArr(k, j + 1) +
InputArr(k + 1, j + 1) - (InputArr(k - 1, j - 1) + 2 * InputArr(k, j - 1) +
InputArr(k + 1, j - 1)))
                    EdgeOut(k, j) = Gx + Gy
            End If
        Next k
    Next j
```

```
End Sub
```

SpatialAnalysis

Qualifiers: Public

```
Sub SpatialAnalysis()
'Finish 2-2-97 sunday hsieh

    Dim I, j, k As Integer
    Dim ArraySize As Integer
    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutputFile

'10-28-96 hsieh
    Dim position() As Single
    Dim M() As Integer
    Dim C() As Integer
    Dim G() As Integer
    Dim L() As Integer
    Dim E() As Integer
    Dim O() As Integer
    Dim S() As Integer                      '11-4-96 hsieh - spatial array
    Dim NewRadArr() As Single
    Dim OriginRadArr() As Single
    Dim EdgeThres    As Single
    Dim xindex, yindex As Integer
    Dim counter, counter_all As Integer
    Dim CompCount() As Integer
    Dim multiplied() As Single
    Dim ECounter, Iteration As Integer
    Dim OriginalStart, OriginalEnd, NewStart, NewEnd As Integer

    Call ReadRadInput(ArraySize, OriginRadArr())
    Call PaddingRadArr(ArraySize, OriginRadArr(), RadArr())

    OriginalStart = 0
    OriginalEnd = ArraySize + 1
    NewStart = 1
    NewEnd = ArraySize

    ReDim VoxelId(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As
Integer
    ReDim EdgeArr(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As
Single
    ReDim NewRadArr(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As
Single

    ReDim M(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As Integer
    ReDim C(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As Integer
    ReDim G(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As Integer
    ReDim L(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As Integer
    ReDim E(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As Integer
    ReDim O(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As Integer
    ReDim S(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As Integer

    '   Reduce 1 voxel width at each side of the RadArr to accommodate
    '   the later processings
'        ArraySize = ArraySize - 2

    ReDim position(1 To 8) As Single
'including un-decided voxels (= CompNum+1   elements)
    ReDim CompCount(0 To CompNum) As Integer

    ReDim multiplied(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As
Single

    Dim TempVoxel()  As Integer
    ReDim TempVoxel(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd) As
Integer
```

```
    Dim TempEdgeVoxel()  As Integer
    ReDim TempEdgeVoxel(OriginalStart To OriginalEnd, OriginalStart To OriginalEnd)
As Integer

    Dim EdgeVoxelCounter, MainVoxelCounter, SpatialVoxelCounter As Integer

'12-4-96 hsieh -------------------------------------
'     Call VariogramSemivariance(InputFile, ".VAR", ArraySize, RadArr())
'---------------------------------------------------

'   Select Exponential data format or Linear data format
    If DataFormOption Then
        For j = OriginalStart To OriginalEnd
            For k = OriginalStart To OriginalEnd
                NewRadArr(k, j) = Exp(RadArr(k, j))
            Next k
        Next j
        EdgeThres = SpatialEdgeThres                          'need to be quantified
    Else
        For j = OriginalStart To OriginalEnd
            For k = OriginalStart To OriginalEnd
                NewRadArr(k, j) = RadArr(k, j)
            Next k
        Next j
        EdgeThres = SpatialEdgeThres                          'need to be quantified
    End If


' 10-28-96 hsieh
' 11-03-96 sunday hsieh

'   1. add phase I~IV remark and change the edge detection to the Phase I
'   2. procesure is compliance with Paper#2 description
' 2-2-97 sunday hsieh
' re-define processings

'   PHASE 0    ----------------
'   temporarily design to identify any single void voxels in space
    ReDim UpperBound(CompNum) As Single
    ReDim LowerBound(CompNum) As Single

    For I = 0 To CompNum - 1
        UpperBound(I) = ComptParameter(0, I) + SpatialStdDevNum * ComptParameter(1,
I)   '1-30-97 hsieh
        LowerBound(I) = ComptParameter(0, I) - SpatialStdDevNum * ComptParameter(1,
I)   '1-30-97 hsieh
    Next I

    For j = OriginalStart To OriginalEnd
        For k = OriginalStart To OriginalEnd
                    C(k, j) = 0
                    G(k, j) = 0
                    M(k, j) = 0
                    S(k, j) = 0
        Next k
    Next j

'1112 retry
'013097 remark by hsieh

'   PHASE I    ----------------
'   2.    Edge detection - Separate the voxels located at  edge regions
'           a. the origin data is expoentially transformed to increase the contrast
'           b. the thresholding value is set to 10
'           c. Robers operation
'           d. Sobel operation
'

'   Select Edge detection operator - Roberts or Sobel
    If EdgeOption = 1 Then
        Call Laplacian(NewRadArr(), ArraySize, EdgeArr())
    ElseIf EdgeOption = 2 Then
        Call Roberts(NewRadArr(), ArraySize, EdgeArr())
    ElseIf EdgeOption = 3 Then
        Call Sobel(NewRadArr(), ArraySize, EdgeArr())
```

```
    End If
    Dim count As Integer
    count = 0
    Dim countNT As Integer
    countNT = 0

'   Define edge pixel if EdgeArr() > thresholding value (EdgeThres)

If EdgeOption = 1 Then 'using Laplacian
    For j = OriginalStart To OriginalEnd
        For k = OriginalStart To OriginalEnd
            If EdgeArr(k, j) = 0# Then
                G(k, j) = 1
                count = count + 1
            Else
                countNT = countNT + 1
            End If
        Next k
    Next j
Else    ' using Sobel and Roberts
    For j = OriginalStart To OriginalEnd
        For k = OriginalStart To OriginalEnd
            If EdgeArr(k, j) >= EdgeThres Then
                G(k, j) = 1
            End If
        Next k
    Next j
End If


'   PHASE II    ----------------
'       1. Define the voxels with a density range that is within 1 std dev difference
'           Identify voxels using normal distribution character
'       2. No overlapping between edge and main-body voxels.
'       3. two options can be found in this phase, all gaussian distribution and
gaussian/thresholding
'           options:
'               3-1 . all-gaussian
'               3-2. gaussian/thresholding
'                       If RadArr( ) <= Mean_#N_min ===> C( )=N_min

'       4. define N components
'               4-1.  Mean_comp#I -(1*StdDev)<= RadArr( ) <= Mean_comp#I -(1*StdDev)
'                                           ===> C( )=I
'               4-2.  If RadArr( ) >= Mean_#N_max -(1*StdDev) ===> C( )=N_max

'       5. M( )=1 if the voxel has been classified as main body voxel or C( )<>0

    For I = 0 To CompNum - 1
        For j = OriginalStart To OriginalEnd
            For k = OriginalStart To OriginalEnd
            If HistoType Then
                If I <> CompNum - 1 And I <> 0 Then
                    If G(k, j) = 0 And RadArr(k, j) >= LowerBound(I) And RadArr(k, j)
<= UpperBound(I) Then
                         C(k, j) = I + 1
                        M(k, j) = 1
                    End If
                Else
                    If G(k, j) = 0 And RadArr(k, j) >= LowerBound(I) Then
                        C(k, j) = I + 1
                        M(k, j) = 1
                    End If
                End If
            Else
                If I = 0 Then
                    If RadArr(k, j) <= UpperBound(0) Then
                         C(k, j) = I + 1
                         M(k, j) = 1
                         G(k, j) = 0
                    End If
                ElseIf I <> CompNum - 1 Then
                    If G(k, j) = 0 And RadArr(k, j) >= LowerBound(I) And RadArr(k, j)
<= UpperBound(I) Then
                         C(k, j) = I + 1
                        M(k, j) = 1
```

```
                End If
            Else
                If G(k, j) = 0 And RadArr(k, j) >= LowerBound(I) Then
                    C(k, j) = I + 1
                    M(k, j) = 1
                End If
            End If
        End If
        Next k
    Next j
Next I

'   PHASE III    ----------------
'       1.  Spatial Voxel Correlation
'       Identify each non-determined voxel, that is M()=0 and G() =0, by comparing
with its 8 surrounding voxels.
'       The character of the voxel is based on the majority character of the
surrounding voxels.
'       For corner voxels, surrounding circumstance is based on
'       the true situation.


    For j = OriginalStart To OriginalEnd
        For k = OriginalStart To OriginalEnd
            If (M(k, j) = 0 And G(k, j) = 0) Then
                S(k, j) = 1
                TempVoxel(k, j) = 1
            Else
                TempVoxel(k, j) = 0
            End If
        Next k
    Next j

    For j = OriginalStart To OriginalEnd
        For k = OriginalStart To OriginalEnd
                If (G(k, j) = 1) Then
                    TempEdgeVoxel(k, j) = 1
                Else
                    TempEdgeVoxel(k, j) = 0
                End If
        Next k
    Next j

    EdgeVoxelCounter = 0
    MainVoxelCounter = 0
    SpatialVoxelCounter = 0
    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
            If S(k, j) = 1 Then
                SpatialVoxelCounter = SpatialVoxelCounter + 1
            ElseIf M(k, j) = 1 Then
                MainVoxelCounter = MainVoxelCounter + 1
             ElseIf G(k, j) = 1 Then
                EdgeVoxelCounter = EdgeVoxelCounter + 1
            End If
        Next k
    Next j

    Dim SIteration, SCounter As Integer
    Dim OldSCounter As Integer
    Dim OldECounter As Integer
    Dim II As Integer
    Dim max, index As Integer
    Dim MinDeltaDensity As Single
    Dim DeltaDensity() As Single
    ReDim DeltaDensity(1 To 8) As Single
    Dim InputArr() As Single
    ReDim InputArr(1 To 8) As Single

    DoEvents
    SCounter = 1
    ECounter = 1
    OldSCounter = 0
    OldECounter = 0

    Iteration = 0
```

```
    SIteration = 0
Dim SumComp As Integer
Dim tempC As Integer

While SCounter <> 0 Or ECounter <> 0
    While SCounter <> 0 And OldSCounter <> SCounter
    OldSCounter = SCounter
    SCounter = 0
    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
            If (TempVoxel(k, j) = 1) Then
                For I = 0 To CompNum
                    CompCount(I) = 0
                Next I
                If k <> NewStart And k <> NewEnd And j <> NewStart And j <> NewEnd
Then ' center portion
                    position(1) = C(k - 1, j - 1)
                    position(2) = C(k, j - 1)
                    position(3) = C(k + 1, j - 1)
                    position(4) = C(k - 1, j)
                    position(5) = C(k + 1, j)
                    position(6) = C(k - 1, j + 1)
                    position(7) = C(k, j + 1)
                    position(8) = C(k + 1, j + 1)
                ElseIf k <> NewStart And k <> NewEnd And j = NewStart Then      'left
column
                    position(1) = 0
                    position(2) = 0
                    position(3) = 0
                    position(4) = C(k - 1, j)
                    position(5) = C(k + 1, j)
                    position(6) = C(k - 1, j + 1)
                    position(7) = C(k, j + 1)
                    position(8) = C(k + 1, j + 1)
                ElseIf k <> NewStart And k <> NewEnd And j = NewEnd Then
'right column
                    position(1) = C(k - 1, j - 1)
                    position(2) = C(k, j - 1)
                    position(3) = C(k + 1, j - 1)
                    position(4) = C(k - 1, j)
                    position(5) = C(k + 1, j)
                    position(6) = 0
                    position(7) = 0
                    position(8) = 0
                ElseIf k = NewStart And j <> NewStart And j <> NewEnd Then      'top
row
                    position(1) = 0
                    position(2) = C(k, j - 1)
                    position(3) = C(k + 1, j - 1)
                    position(4) = 0
                    position(5) = C(k + 1, j)
                    position(6) = 0
                    position(7) = C(k, j + 1)
                    position(8) = C(k + 1, j + 1)
                ElseIf k = NewEnd And j <> NewStart And j <> NewEnd Then
'bottom row
                    position(1) = C(k - 1, j - 1)
                    position(2) = C(k, j - 1)
                    position(3) = 0
                    position(4) = C(k - 1, j)
                    position(5) = 0
                    position(6) = C(k - 1, j + 1)
                    position(7) = C(k, j + 1)
                    position(8) = 0
                ElseIf k = NewStart And j = NewStart Then         'upper-left corner
                    position(1) = 0
                    position(2) = 0
                    position(3) = 0
                    position(4) = 0
                    position(5) = C(k + 1, j)
                    position(6) = 0
                    position(7) = C(k, j + 1)
                    position(8) = C(k + 1, j + 1)
                ElseIf k = NewStart And j = NewEnd Then         'upper-right corner
                    position(1) = 0
                    position(2) = C(k, j - 1)
```

```
                    position(3) = C(k + 1, j - 1)
                    position(4) = 0
                    position(5) = C(k + 1, j)
                    position(6) = 0
                    position(7) = 0
                    position(8) = 0
                ElseIf k = NewEnd And j = NewStart Then          'lower-left corner
                    position(1) = 0
                    position(2) = 0
                    position(3) = 0
                    position(4) = C(k - 1, j)
                    position(5) = 0
                    position(6) = C(k - 1, j + 1)
                    position(7) = C(k, j + 1)
                    position(8) = 0
                ElseIf k = NewEnd And j = NewEnd Then          'lower-right corner
                    position(1) = C(k - 1, j - 1)
                    position(2) = C(k, j - 1)
                    position(3) = 0
                    position(4) = C(k - 1, j)
                    position(5) = 0
                    position(6) = 0
                    position(7) = 0
                    position(8) = 0
                End If

                For I = 0 To CompNum
                    For II = 1 To 8
                        If position(II) = I Then
                            CompCount(I) = CompCount(I) + 1
                        End If
                    Next II
                Next I
'change to function ====================================
                max = 0
                index = 0

                For I = 1 To CompNum
                    If CompCount(I) > max Then
                        max = CompCount(I)
                        index = I
                    End If
                Next I
                If index <> 0 And max > 1 Then
                        C(k, j) = index
                        TempVoxel(k, j) = 0
                End If
'change to function ====================================
            End If
        Next k
    Next j

    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
            If TempVoxel(k, j) = 1 Then
                SCounter = SCounter + 1
            End If
        Next k
    Next j
    SIteration = SIteration + 1
Wend
OldSCounter = 0


'   PHASE IV    ----------------
'       1.  Edge pixel correction - redefine the voxel density at edge by the
gradient
    While (ECounter) <> 0 And OldECounter <> ECounter
    OldECounter = ECounter
    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
            If (TempEdgeVoxel(k, j) = 1) Then
'===============================
    If k <> NewStart And k <> NewEnd And j <> NewStart And j <> NewEnd Then ' center
portion
```

```
    InputArr(1) = NewRadArr(k - 1, j - 1)
    InputArr(2) = NewRadArr(k, j - 1)
    InputArr(3) = NewRadArr(k + 1, j - 1)
    InputArr(4) = NewRadArr(k - 1, j)
    InputArr(5) = NewRadArr(k + 1, j)
    InputArr(6) = NewRadArr(k - 1, j + 1)
    InputArr(7) = NewRadArr(k, j + 1)
    InputArr(8) = NewRadArr(k + 1, j + 1)
ElseIf k <> NewStart And k <> NewEnd And j = NewStart Then      'left column
    InputArr(1) = 1000#
    InputArr(2) = 1000#
    InputArr(3) = 1000#
    InputArr(4) = NewRadArr(k - 1, j)
    InputArr(5) = NewRadArr(k + 1, j)
    InputArr(6) = NewRadArr(k - 1, j + 1)
    InputArr(7) = NewRadArr(k, j + 1)
    InputArr(8) = NewRadArr(k + 1, j + 1)
ElseIf k <> NewStart And k <> NewEnd And j = NewEnd Then            'right column
    InputArr(1) = NewRadArr(k - 1, j - 1)
    InputArr(2) = NewRadArr(k, j - 1)
    InputArr(3) = NewRadArr(k + 1, j - 1)
    InputArr(4) = NewRadArr(k - 1, j)
    InputArr(5) = NewRadArr(k + 1, j)
    InputArr(6) = 1000#
    InputArr(7) = 1000#
    InputArr(8) = 1000#
ElseIf k = NewStart And j <> NewStart And j <> NewEnd Then      'top  row
    InputArr(1) = 1000#
    InputArr(2) = NewRadArr(k, j - 1)
    InputArr(3) = NewRadArr(k + 1, j - 1)
    InputArr(4) = 1000#
    InputArr(5) = NewRadArr(k + 1, j)
    InputArr(6) = 1000#
    InputArr(7) = NewRadArr(k, j + 1)
    InputArr(8) = NewRadArr(k + 1, j + 1)
ElseIf k = NewEnd And j <> NewStart And j <> NewEnd Then         'bottom row
    InputArr(1) = NewRadArr(k - 1, j - 1)
    InputArr(2) = NewRadArr(k, j - 1)
    InputArr(3) = 1000#
    InputArr(4) = NewRadArr(k - 1, j)
    InputArr(5) = 1000#
    InputArr(6) = NewRadArr(k - 1, j + 1)
    InputArr(7) = NewRadArr(k, j + 1)
    InputArr(8) = 1000#
ElseIf k = NewStart And j = NewStart Then         'upper-left corner
    InputArr(1) = 1000#
    InputArr(2) = 1000#
    InputArr(3) = 1000#
    InputArr(4) = 1000#
    InputArr(5) = NewRadArr(k + 1, j)
    InputArr(6) = 1000#
    InputArr(7) = NewRadArr(k, j + 1)
    InputArr(8) = NewRadArr(k + 1, j + 1)
ElseIf k = NewStart And j = NewEnd Then         'upper-right corner
    InputArr(1) = 1000#
    InputArr(2) = NewRadArr(k, j - 1)
    InputArr(3) = NewRadArr(k + 1, j - 1)
    InputArr(4) = 1000#
    InputArr(5) = NewRadArr(k + 1, j)
    InputArr(6) = 1000#
    InputArr(7) = 1000#
    InputArr(8) = 1000#
ElseIf k = NewEnd And j = NewStart Then         'lower-left corner
    InputArr(1) = 1000#
    InputArr(2) = 1000#
    InputArr(3) = 1000#
    InputArr(4) = NewRadArr(k - 1, j)
    InputArr(5) = 1000#
    InputArr(6) = NewRadArr(k - 1, j + 1)
    InputArr(7) = NewRadArr(k, j + 1)
    InputArr(8) = 1000#
ElseIf k = NewEnd And j = NewEnd Then         'lower-right corner
    InputArr(1) = NewRadArr(k - 1, j - 1)
    InputArr(2) = NewRadArr(k, j - 1)
    InputArr(3) = 1000#
    InputArr(4) = NewRadArr(k - 1, j)
```

```
        InputArr(5) = 1000#
        InputArr(6) = 1000#
        InputArr(7) = 1000#
        InputArr(8) = 1000#
    End If

        position(1) = C(k - 1, j - 1)
        position(2) = C(k, j - 1)
        position(3) = C(k + 1, j - 1)
        position(4) = C(k - 1, j)
        position(5) = C(k + 1, j)
        position(6) = C(k - 1, j + 1)
        position(7) = C(k, j + 1)
        position(8) = C(k + 1, j + 1)

'   Search for the minimun gradient between RadArr(k,j) and its surrounded 8 voxels -
-
    For I = 1 To 8
        DeltaDensity(I) = Abs(InputArr(I) - NewRadArr(k, j))
    Next I

    MinDeltaDensity = 100#
    index = 9
    For I = 1 To 8
       If (position(I) <> 0) And (DeltaDensity(I) < MinDeltaDensity) Then
            MinDeltaDensity = DeltaDensity(I)
            index = I
       End If
    Next I

                Select Case index
                Case 1:
                        C(k, j) = C(k - 1, j - 1)
                Case 2:
                        C(k, j) = C(k, j - 1)
                Case 3:
                        C(k, j) = C(k + 1, j - 1)
                Case 4:
                        C(k, j) = C(k - 1, j)
                Case 5:
                        C(k, j) = C(k + 1, j)
                Case 6:
                        C(k, j) = C(k - 1, j + 1)
                Case 7:
                        C(k, j) = C(k, j + 1)
                Case 8:
                        C(k, j) = C(k + 1, j + 1)
                Case 9:
                        GoTo Try_Again
                End Select

                If C(k, j) <> 0 Then
                    TempEdgeVoxel(k, j) = 0
                End If
             End If
Try_Again:
        Next k
    Next j

    ECounter = 0
    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
            If TempEdgeVoxel(k, j) = 1 Then
                ECounter = ECounter + 1
            End If
        Next k
    Next j
        Iteration = Iteration + 1
    Wend
    OldECounter = 0

    If SIteration > 5000 Then
        For j = NewStart To NewEnd
            For k = NewStart To NewEnd
                If TempEdgeVoxel(k, j) = 1 Then
                    TempEdgeVoxel(k, j) = 0
```

```
                C(k, j) = 0
            ElseIf TempVoxel(k, j) = 1 Then
                TempVoxel(k, j) = 0
                C(k, j) = 0
            End If
        Next k
    Next j
End If

Wend

'   11-5-96    hsieh
'   PHASE V    ----------------
'       1. Calculate the percentage of each componet which is defined by
'           3 segregation processings - MainBody, SpatialCorr, Edge
Dim EdgeVoxels() As Single
Dim MainVoxels() As Single
Dim SpatialVoxels() As Single
ReDim EdgeVoxels(NewStart To NewEnd, NewStart To NewEnd) As Single
ReDim MainVoxels(NewStart To NewEnd, NewStart To NewEnd) As Single
ReDim SpatialVoxels(NewStart To NewEnd, NewStart To NewEnd) As Single

Dim CompVoxel() As Single
ReDim CompVoxel(NewStart To NewEnd, NewStart To NewEnd, 0 To CompNum - 1) As Single

    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
            EdgeVoxels(k, j) = G(k, j) * RadArr(k, j) / Scaling
            MainVoxels(k, j) = M(k, j) * RadArr(k, j) / Scaling
            SpatialVoxels(k, j) = S(k, j) * RadArr(k, j) / Scaling
            RadArr(k, j) = RadArr(k, j) / Scaling
        Next k
    Next j

    For I = 0 To CompNum - 1
        For j = NewStart To NewEnd
            For k = NewStart To NewEnd
                If C(k, j) = I + 1 Then
                    CompVoxel(k, j, I) = RadArr(k, j)
                Else
                    CompVoxel(k, j, I) = 0#
                End If
            Next k
        Next j
    Next I

'   10-28-96    hsieh
'   Write results to the specified output files with distinct extension name

'   1.  Component Image - C()
    Call WriteOutputInteger(InputFile, "C.SPA", ArraySize, C())
'   2.  Edge operation results - Edge()
    Call WriteOutputSingle(InputFile, "E.SPA", ArraySize, EdgeArr())
'   3.  Edge determined after thresholding - G()
    Call WriteOutputInteger(InputFile, "G.SPA", ArraySize, G())
'   4.  Main body voxels - M()
   Call WriteOutputInteger(InputFile, "M.SPA", ArraySize, M())
'   5.  Voxel at Space - S()
   Call WriteOutputInteger(InputFile, "S.SPA", ArraySize, S())

'   6.  Edge operation results - Edge()
'    Call WriteOutputSingle(InputFile, "E.VOL", ArraySize, EdgeVoxels())
'   7.  Edge operation results - Edge()
'    Call WriteOutputSingle(InputFile, "M.VOL", ArraySize, MainVoxels())
'   8.  Edge operation results - Edge()
'    Call WriteOutputSingle(InputFile, "S.VOL", ArraySize, SpatialVoxels())
'   9.  Summary results used for Excel calculation
'    Call WriteSummaryOutput(InputFile, ".OUT", "Main", "Spatial", "Edge", ArraySize,
NewRadArr(), MainVoxels(), SpatialVoxels(), EdgeVoxels())
'  10.  Summary results used for Excel calculation
    Call WriteCompSummary(InputFile, ".OUT", ArraySize, CompNum, RadArr(),
CompVoxel())


'   9-26-96 hseih ------------------------------------------------------------------
-----------------------
```

```
'   calculate the percentage of each component

Dim IndexComp() As Integer
ReDim IndexComp(0 To CompNum) As Integer

Dim IndexSum() As Integer
ReDim IndexSum(0 To CompNum) As Integer

Dim TotalVoxel As Single

For I = 0 To CompNum
    IndexComp(I) = 0
Next I

TotalVoxel = NewEnd * NewEnd
For I = 0 To CompNum - 1
    For j = NewStart To NewEnd
        For k = NewStart To NewEnd
            If C(k, j) = I + 1 Then
                IndexComp(I) = IndexComp(I) + 1
            End If
        Next k
    Next j
Next I

    For I = 0 To CompNum - 1
        IndexSum(I) = IndexComp(I) / TotalVoxel * 1#
    Next I

MsgBox "Procedure Completed!"

End Sub
```

StartProcessing

Qualifiers: Public

```
Sub StartProcessing()
    Call DoCalculations
    Call WriteOutputData
    MsgBox "Process Complete"
End Sub
```

VariogramParameters

Qualifiers: Public

Arguments:	Size	Integer	By Value
	SepDist	Single	By Ref.
	SemiArr	Single	By Ref.
	SemiPara	Double	By Ref.

```
Sub VariogramParameters(ByVal Size As Integer, SepDist() As Single, SemiArr() As
Single, SemiPara() As Double)
    ' Calculated the following:
        '       1. sill,
        '       2. nugget,
        '       3. slope of curve,
        '       4. the curve's intercept,
        '       5. r^2 values,
        '       6. the range.

        '       SemiPara(j,1)= Var_Sill
        '       SemiPara(j,2)= Nugget
        '       SemiPara(j,3)= Slope
        '       SemiPara(j,4)= Intercept
```

```
        '       SemiPara(j,5)= RSquare
        '       SemiPara(j,6)= Range

    '   2 priori information are used to deal with some special cases
    '       If the slope used for determining Nugget and Intercept values goes to
INFINITY,
    '       the Slope  is set to 100000 and
    '       both Nugget and Intercept values become LARGE and NEGATIVE
    '       and the Range value becomes a SMALL, POSITIVE value.

ReDim SemiPara(1 To Size, 1 To 6) As Double
Dim I, j, k As Integer
Dim Cycle, Calc As Integer
Dim var_dif, var1, var2 As Single
Dim total, sum_x_sq, sum_xy, sum_x, sum_y, sum_xx, sum_yy As Single
Dim ybar, xbar As Single
Dim M As Double
Dim datapts, HalfSize As Integer

HalfSize = Int(Size / 2)
For j = 1 To Size
    sum_x_sq = 0#
    sum_xy = 0#
    sum_x = 0#
    sum_y = 0#
    sum_xx = 0#
    sum_yy = 0#

    For k = 1 To HalfSize
        sum_x = SemiArr(k, j) + sum_x
        sum_x_sq = SemiArr(k, j) * SemiArr(k, j) + sum_x_sq
    Next k

'Calculate variance for entire data set.
    var1 = ((sum_x_sq - (sum_x * sum_x) / HalfSize) / (HalfSize - 1))
    sum_x = 0#
    sum_x_sq = 0#
    var2 = 0#
    Cycle = 1
    var_dif = var1

'Locate distance at which variance is at the first minimum.
    Do
        Calc = Int(HalfSize - Cycle)
        Cycle = Cycle + 1
        For k = Cycle To HalfSize
            sum_x = SemiArr(k, j) + sum_x
            sum_x_sq = SemiArr(k, j) * SemiArr(k, j) + sum_x_sq
        Next k
        var2 = ((sum_x_sq - (sum_x * sum_x) / Calc) / (Calc - 1))
        sum_x = 0#
        sum_x_sq = 0#
        var_dif = var1 - var2
        var1 = var2
    Loop Until var_dif <= 0

'Calculate Mean of remaining Points - Sill Semivariance.
 'SemiPara(j,1)= Sill
    total = 0#
    For k = Cycle To HalfSize
        total = total + SemiArr(k, j)
    Next k
    SemiPara(j, 1) = total / (HalfSize - (Cycle - 1))

'Calculate Nugget = SemiPara(j,2)
'This calculation is performed using data rejected for sill.
    sum_xy = 0#
    sum_x = 0#
    sum_y = 0#
    sum_xx = 0#
    sum_yy = 0#
    datapts = Cycle - 1

    For k = 1 To datapts
        sum_xy = SemiArr(k, j) * SepDist(k) + sum_xy
        sum_x = SepDist(k) + sum_x
```

```
        sum_y = SemiArr(k, j) + sum_y
        sum_xx = SepDist(k) * SepDist(k) + sum_xx
        sum_yy = SemiArr(k, j) * SemiArr(k, j) + sum_yy
    Next k

    ' priori information 1
    If (sum_xx - (sum_x * sum_x)) = 0# Then
        M = 100000000#
    Else
        M = (sum_xy - (sum_x * sum_y / datapts)) / (sum_xx - (sum_x * sum_x) /
datapts)
    End If
    ybar = sum_y / datapts
    xbar = sum_x / datapts
    SemiPara(j, 2) = ybar - M * xbar                                'SemiPara(j,2)=
Nugget

    sum_xy = 0#
    sum_x = 0#
    sum_y = 0#
    sum_xx = 0#
    sum_yy = 0#
    k = 1
    Do
        sum_xy = SemiArr(k, j) * SepDist(k) + sum_xy
        sum_x = SepDist(k) + sum_x
        sum_y = SemiArr(k, j) + sum_y
        sum_xx = SepDist(k) * SepDist(k) + sum_xx
        sum_yy = SemiArr(k, j) * SemiArr(k, j) + sum_yy
        k = k + 1
    Loop Until SemiArr(k, j) > SemiPara(j, 1)
    datapts = k - 1
    ybar = sum_y / datapts
    xbar = sum_x / datapts

'SemiPara(j,3)= Slope
' priori information 2
    If (sum_xx - (sum_x * sum_x)) = 0# Then
        SemiPara(j, 3) = 100000000#
    Else
        SemiPara(j, 3) = (sum_xy - (sum_x * sum_y / datapts)) / (sum_xx - (sum_x *
sum_x) / datapts)
    End If
'SemiPara(j,4)= Intercept
    SemiPara(j, 4) = ybar - SemiPara(j, 3) * xbar
'SemiPara(j,5)= R-Square
    SemiPara(j, 5) = (sum_xy * sum_xy) / (sum_xx * sum_yy)
'SemiPara(j,6)= Range
    SemiPara(j, 6) = (SemiPara(j, 1) - SemiPara(j, 4)) / SemiPara(j, 3)
Next j

End Sub
```

VariogramSemivariance

Qualifiers: Public

```
Sub VariogramSemivariance()
    ' 12-5-96 Hsieh Env. Support Lab Rm#105
    ' Modified from Dr. Solie's Excel Macro program -SEMIMACR.XLS  (recorded 9/7/96 )

    ' Three sub-routines are written: VariogramSemivariance, VariogramParameters and
WriteVariogram

    ' 0. Input file format - start with array size (N) and then NxN data points in
the same row.
    '   eg:   68
    '           0.0189
    '           0.0185
    '           0.0184
    '           0.0196
```

```
    '         :
    '         :
    '   All *.RAD files are OK to use.

    '1. VariogramSemivariance
        ' Calculate semivariorgrams for a NxN data array
        ' Apply on ONE-direction only (saying x-direction).
        '          It would be easy to sample y-direction by some minor change.
        ' The output data consist of two columns - the distance and semivariance
arrays
        '
^^^^^^^^^       ^^^^^^^^^^^^

    ' 2. VariogramParameters
        ' Calculated the following:
        '       1. sill,
        '       2. nugget,
        '       3. slope of curve,
        '       4. the curve's intercept,
        '       5. r^2 values,
        '       6. the range.

    ' 3. WriteVariogram
        ' It generates N files with extension name of ".VAR".
        ' Each file consists of (3) columns x (N) rows - row data ,distance,
semivariance
        ' Though the resulting array of the semivariance array should be 1/2 N,
        '       for the ease of writting output, it is also has a size of N elements


    Dim I, j, k As Integer
    Dim Size As Integer
    Dim Calc, Cycle, RowNum, SemiIncr As Integer
    Dim dist, delta As Single
    Dim dif, sumdif, sumdifsq As Single
    Dim SepDist() As Single
    Dim Semivar() As Single
    Dim SemiPara() As Double

    Call ReadRadInput(Size, RadArr())

    ReDim SepDist(1 To Size) As Single
    ReDim Semivar(1 To Size, 1 To Size) As Single
    ReDim SemiPara(1 To Size, 1 To 6) As Double

    dist = Size * 1.5
    RowNum = Size
    delta = dist / RowNum


For j = 1 To Size
    SemiIncr = Int(RowNum / 2)
    Cycle = 0
    Calc = 0

    dif = 0#
    sumdif = 0#
    sumdifsq = 0#

        For k = 1 To SemiIncr
            Cycle = 1 + Cycle
            Calc = Int(RowNum - Cycle)
            For I = 1 To Calc
                dif = RadArr(I + Cycle, j) - RadArr(I, j) ' row (=k) 1 to 68
                'dif = RadArr(j, I + Cycle) - RadArr(j, I)   ' col   (=j)  1 to 68
                sumdif = dif + sumdif
                sumdifsq = dif * dif + sumdifsq
            Next I
            SepDist(j) = j * delta
            Semivar(k, j) = ((sumdifsq - (sumdif * sumdif) / Calc) / (Calc - 1)) / 2
            sumdif = 0
            sumdifsq = 0
        Next k
Next j
    Call VariogramParameters(Size, SepDist(), Semivar(), SemiPara())
```

```
    Call WriteVariogram(InputFile, ".VAR", Size, RadArr(), Semivar(), SepDist(),
SemiPara())

    MsgBox "Procedure Completed!"

End Sub
```

WriteCompSummary

Qualifiers:	Public		
Arguments:	InFile	Variant	By Ref.
	Size	Integer	By Value
	Cnum	Integer	By Value
	Rad	Single	By Ref.
	temp	Single	By Ref.

```
Sub WriteCompSummary(InFile, ExtName, ByVal Size As Integer, ByVal Cnum As Integer,
Rad() As Single, temp() As Single)
    Dim I, j, k As Integer
    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutFile
    Dim FileNum

    OutFileNum = FreeFile
    StrLen = Len(InFile)
    OutFile = Mid$(InFile, 1, StrLen - 4)
    OutFile = OutFile & ExtName
    Open OutFile For Output As #OutFileNum
        Dim temp1() As Single
        Dim temp2() As Single
        Dim temp3() As Single

        ReDim temp1(1 To Size, 1 To Size) As Single
        ReDim temp2(1 To Size, 1 To Size) As Single
        ReDim temp3(1 To Size, 1 To Size) As Single

    For I = 0 To Cnum - 1
        For j = 1 To Size
            For k = 1 To Size
                    If I = 0 Then
                        temp1(k, j) = temp(k, j, I)
                    ElseIf I = 1 Then
                        temp2(k, j) = temp(k, j, I)
                    ElseIf I = 2 Then
                        temp3(k, j) = temp(k, j, I)
                    End If
            Next k
        Next j
    Next I

    Print #OutFileNum, "Origin", "Comp1", "Comp2", "Comp3"
        For j = 1 To Size
            For k = 1 To Size
                Print #OutFileNum, Format(Rad(k, j), "##0.0000"), Format(temp1(k, j),
"##0.0000"), Format(temp2(k, j), "##0.0000"), Format(temp3(k, j), "##0.0000")
            Next k
        Next j

    Close #OutFileNum

End Sub
```

WriteLaplaceOutput

Qualifiers: Public
Arguments: LaplaceFile String By Ref.

```
Sub WriteLaplaceOutput(LaplaceFile As String)
    'This routine uses the global variable NV and the
    'global array Mu (which holds the results of the
    'computations)

    'generate Laplace opertor approximation array

    Dim Ix As Integer, Iy As Integer
    Dim FileNum
    Dim Mu_e() As Single
    ReDim Mu_e(1 To NV, 1 To NV)


    FileNum = FreeFile
    Open LaplaceFile For Output As #FileNum

    Print #FileNum, NV
    For Iy = 1 To NV
        For Ix = 1 To NV
            If (Iy = 1) Then
                Mu_e(Ix, Iy) = 1# / (1# + 136# * Abs(Mu(Ix, Iy) - Mu(Ix, Iy + 1)))
            ElseIf (Ix = 1) Then
                Mu_e(Ix, Iy) = 1# / (1# + 136# * Abs(Mu(Ix, Iy) - Mu(Ix + 1, Iy)))
            ElseIf (Iy = NV) Then
                Mu_e(Ix, Iy) = 1# / (1# + 136# * Abs(Mu(Ix, Iy) - Mu(Ix, Iy - 1)))
            ElseIf (Ix = NV) Then
                Mu_e(Ix, Iy) = 1# / (1# + 136# * Abs(Mu(Ix, Iy) - Mu(Ix - 1, Iy)))
            Else
                Mu_e(Ix, Iy) = 1# / (1# + 136# * Abs(Mu(Ix, Iy) - 0.25 * (Mu(Ix, Iy -
1) + Mu(Ix - 1, Iy) + Mu(Ix + 1, Iy) + Mu(Ix, Iy + 1))))
            End If
                'Mu_e(Ix, Iy) = Mu_e(Ix, Iy) * Mu(Ix, Iy)

        Next Ix
    Next Iy
    For Iy = 1 To NV
        For Ix = 1 To NV
            If (Iy >= 28 And Iy <= 94) And (Ix >= 28 And Ix <= 94) Then
                'Print #FileNum, Mu(Ix, Iy)
                Print #FileNum, Mu_e(Ix, Iy)
            End If
        Next Ix
    Next Iy

    Close #FileNum

End Sub
```

WriteOutputData

Qualifiers: Public

```
Sub WriteOutputData()
    Dim StrLen, OutputFile

    StrLen = Len(InputFile)
    OutputFile = Mid$(InputFile, 1, StrLen - 4)

    'First we write the processed data onto a file with an
    'extension .RAD

    'add 10-8-95 for generating Laplace opertor approximation ---- at home ----
    'Call WriteLaplaceOutput(OutputFile & "h.RAD")

    If frmRadon.chkRadOutput.Value = True Then
        Debug.Print "Writing Radon Output File"
```

```
        Call WriteRadonOutput(OutputFile & ".RAD")
    End If

    If frmRadon.chkRadError.Value = True Then
        Debug.Print "Writing Radon Error File"
        Call WriteRadonError(OutputFile & ".ERR")
    End If

    'If frmRadon.chkSurferOutput.Value = True Then
    '    WriteSurferOutput (OutputFile & ".GRD")
    'End If

End Sub
```

WriteOutputInteger

Qualifiers:	Public		
Arguments:	InFile	Variant	By Ref.
	ExtName	String	By Ref.
	Size	Integer	By Ref.
	TempArray	Integer	By Ref.

```
Sub WriteOutputInteger(InFile, ExtName As String, Size As Integer, TempArray() As
Integer)

Dim j, k As Integer
    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutFile
    Dim FileNum

        OutFileNum = FreeFile
        StrLen = Len(InFile)
        OutFile = Mid$(InFile, 1, StrLen - 4)
        OutFile = OutFile & ExtName
        Open OutFile For Output As #OutFileNum

        Print #OutFileNum, Size
        For j = 1 To Size
            For k = 1 To Size
                Print #OutFileNum, TempArray(k, j)
            Next k
        Next j
        Close #OutFileNum

End Sub
```

WriteOutputSingle

Qualifiers:	Public		
Arguments:	InFile	Variant	By Ref.
	ExtName	String	By Ref.
	Size	Integer	By Ref.
	TempArray	Single	By Ref.

```
Sub WriteOutputSingle(InFile, ExtName As String, Size As Integer, TempArray() As
Single)
    Dim j, k As Integer
    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutFile
    Dim FileNum

    OutFileNum = FreeFile
    StrLen = Len(InFile)
```

```
    OutFile = Mid$(InFile, 1, StrLen - 4)
    OutFile = OutFile & ExtName
    Open OutFile For Output As #OutFileNum

    Print #OutFileNum, Size
    For j = 1 To Size
        For k = 1 To Size
            Print #OutFileNum, TempArray(k, j)
        Next k
    Next j
    Close #OutFileNum

End Sub
```

WriteRadonError

Qualifiers: Public
Arguments: RadonErrorFile String By Ref.

```
Sub WriteRadonError(RadonErrorFile As String)
    'This routine uses the global variable NV and the
    'global array Mu (which holds the results of the
    'computations)
    Dim Ix As Integer, Iy As Integer
    Dim FileNum

    FileNum = FreeFile
    Open RadonErrorFile For Output As #FileNum

    Print #FileNum, NV
    For Iy = 1 To NV
        For Ix = 1 To NV
            Print #FileNum, Sigma(Ix, Iy)
        Next Ix
    Next Iy

    Close #FileNum

End Sub
```

WriteRadonOutput

Qualifiers: Public
Arguments: RadonOutputFile String By Ref.

```
Sub WriteRadonOutput(RadonOutputFile As String)
    'This routine uses the global variable NV and the
    'global array Mu (which holds the results of the
    'computations)
    Dim Ix As Integer, Iy As Integer
    Dim FileNum

    FileNum = FreeFile
    Open RadonOutputFile For Output As #FileNum

'=========================================================
'temp processing - generate spectrum 6-30-96 hsieh
    'For Iy = 1 To NV
    '    For Ix = 1 To NV
    '        If Iy <= 5 And Ix <= 60 Then
    '            Mu(Iy, Ix) = -.002 + .002 * Int((Ix - 1) / 5)
    '        End If
    '    Next Ix
    'Next Iy
'=========================================================
```

```
    Print #FileNum, NV
    For Iy = 1 To NV
        For Ix = 1 To NV
            Print #FileNum, Mu(Ix, Iy)
        Next Ix
    Next Iy

    Close #FileNum

End Sub
```

WriteSummaryComp

Qualifiers:	Public		
Arguments:	InFile	Variant	By Ref.
	ExtName	Variant	By Ref.
	Size	Integer	By Ref.
	temp1	Single	By Ref.
	temp2	Single	By Ref.

```
Sub WriteSummaryComp(InFile, ExtName, Size As Integer, temp1() As Single, temp2() As
Single)
    Dim j, k As Integer
    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutFile
    Dim FileNum

    OutFileNum = FreeFile
    StrLen = Len(InFile)
    OutFile = Mid$(InFile, 1, StrLen - 4)
    OutFile = OutFile & ExtName
    Open OutFile For Output As #OutFileNum

    'For I = 0 To CompNum - 1
    '    temp = "temparray" & I + 1
    '    Dim temp()
    '    ReDim temp(NewStart To NewEnd, NewStart To NewEnd, 0 To CompNum - 1) As
Single
   ' Next I

    'Print #OutFileNum, "Origin", Comp1, Comp2, Comp3
   ' For I = 0 To CompNum - 1
   ' For j = 1 To Size
   '     For k = 1 To Size
   '         Print #OutFileNum, Format(Temp1(k, j) / Scaling, "##0.0000"),
Format(Temp2(k, j, I), "##0.0000")
   '     Next k
   ' Next j
    'Next I
    Close #OutFileNum

End Sub
```

WriteSummaryOutput

Qualifiers:	Public		
Arguments:	InFile	Variant	By Ref.
	ExtName	Variant	By Ref.
	Title1	Variant	By Ref.
	Title2	Variant	By Ref.
	Title3	String	By Ref.
	Size	Integer	By Ref.

temp1	Single	By Ref.
temp2	Single	By Ref.
temp3	Single	By Ref.
Temp4	Single	By Ref.

```
Sub WriteSummaryOutput(InFile, ExtName, Title1, Title2, Title3 As String, Size As
Integer, temp1() As Single, temp2() As Single, temp3() As Single, Temp4() As Single)
    Dim j, k As Integer
    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutFile
    Dim FileNum

    OutFileNum = FreeFile
    StrLen = Len(InFile)
    OutFile = Mid$(InFile, 1, StrLen - 4)
    OutFile = OutFile & ExtName
    Open OutFile For Output As #OutFileNum

    Print #OutFileNum, "Origin", Title1, Title2, Title3
    For j = 1 To Size
        For k = 1 To Size
            Print #OutFileNum, Format(temp1(k, j) / 135.67, "##0.0000"),
Format(temp2(k, j), "##0.0000"), Format(temp3(k, j), "##0.0000"), Format(Temp4(k, j),
"##0.0000")
        Next k
    Next j
    Close #OutFileNum

End Sub
```

WriteSurferOutput

Qualifiers:	Public		
Arguments:	SurferOutputFile	Variant	By Ref.

```
Sub WriteSurferOutput(SurferOutputFile)
    Dim Ix As Integer, Iy As Integer
    Dim Xmin As Integer, Ymin As Integer
    Dim Xmax As Integer, Ymax As Integer
    Dim MinZ As Single, MaxZ As Single
    Dim FileNum

    'This routine uses the global variable NV and the
    'global array Mu (which holds the results of the
    'computations)

    '**** Scaling Factor For 'SURFER' ****
    Xmin = 1
    Xmax = NV 'Currently surfer cannot handle an X-value of 120
    Ymin = 1
    Ymax = NV
    MinZ = MinimumOf2dArrayMu()
    MaxZ = MaximumOf2dArrayMu()
    'MinZ = -.0034
    'MaxZ = .0233

    FileNum = FreeFile
    Open SurferOutputFile For Output As #FileNum

    Print #FileNum, "DSAA"  'DSAA is the signature of a Surfer file
    Print #FileNum, NV / 2; NV
    Print #FileNum, Xmin; Xmax
    Print #FileNum, Ymin; Ymax
    Print #FileNum, MinZ; MaxZ
    For Iy = 1 To NV
        For Ix = 1 To NV 'Currently surfer cannot handle an X-value of 120
            Print #FileNum, Format(Mu(Ix, Iy), "00000000.00000"); " ";
        Next Ix
```

```
        Print #FileNum,
    Next Iy

    Close #FileNum

End Sub
```

WriteVariogram

Qualifiers:	Public		
Arguments:	InFile	Variant	By Ref.
	ExtName	String	By Ref.
	Size	Integer	By Ref.
	RadArr	Single	By Ref.
	Semivar	Single	By Ref.
	SepDist	Single	By Ref.
	SemiPara	Double	By Ref.

```
Sub WriteVariogram(InFile, ExtName As String, Size As Integer, RadArr() As Single,
Semivar() As Single, SepDist() As Single, SemiPara() As Double)
'write output for Semivariogram

    Dim OutFileNum, FileLength, NextLine
    Dim StrLen, OutFile
    Dim FileNum
    Dim I, j, k As Integer

    For j = 1 To Size

        OutFileNum = FreeFile
        StrLen = Len(InFile)
        OutFile = Mid$(InFile, 1, StrLen - 4)
        OutFile = OutFile & j & ExtName
        Open OutFile For Output As #OutFileNum

        For k = 1 To Size
            Print #OutFileNum, Format(RadArr(k, j), "0.000E+00"), Format(SepDist(k),
"##0.00"), Format(Semivar(k, j), "0.000E+00") ' row (=k) 1 to 68
        Next k

        For I = 1 To 6
            Print #OutFileNum, , Format(I, "##0000"), Format(SemiPara(j, I),
"0.000E+00")
        Next I
        Close #OutFileNum

    Next j

End Sub
```

Functions

ComponentNumber

Qualifiers:	Public		
Arguments:	CCount	Integer	By Ref.
	CompNums	Integer	By Ref.
Returns:	Integer		

```
Function ComponentNumber(CCount() As Integer, CompNums As Integer) As Integer
```

```
'return a single number that indicates the type of the component.
'ComponentNumber=3 ===> dolomite
'ComponentNumber=2 ===> gypsum
'ComponentNumber=1 ===> voiud
'ComponentNumber=0 ===> undecided

Dim I, index, max As Integer

max = CCount(0)
index = 1

For I = 1 To CompNums - 1
    If CCount(I) > max Then
        max = CCount(I)
        index = I + 1
    End If
Next I
    ComponentNumber = index
End Function
```

max

Qualifiers:	Public		
Arguments:	A	Single	By Ref.
	B	Single	By Ref.
Returns:	Variant		

```
Function max(A As Single, B As Single)
    max = A
    If A < B Then
        max = B
    End If
End Function
```

MaximumOf2dArrayMu

Qualifiers:	Public
Returns:	Variant

```
Function MaximumOf2dArrayMu()
    Dim I As Integer, j As Integer
    Dim Maximum

    Maximum = Mu(1, 1)
    For I = 1 To NV
        For j = 1 To NV
            If Mu(I, j) > Maximum Then
                Maximum = Mu(I, j)
            End If
        Next j
    Next I

    MaximumOf2dArrayMu = Maximum
End Function
```

Min

Qualifiers:	Public		
Arguments:	A	Single	By Ref.
	B	Single	By Ref.

Returns: Variant

```
Function Min(A As Single, B As Single)

    Min = B
    If A < B Then
        Min = A
    End If
End Function
```

MinDeltaDensityIndex

Qualifiers:	Public		
Arguments:	X	Integer	By Value
	Y	Integer	By Value
	Voxel	Single	By Value
	UD	Integer	By Ref.
	CC	Integer	By Ref.
Returns:	Integer		

```
Function MinDeltaDensityIndex(ByVal X As Integer, ByVal Y As Integer, ByVal Voxel As
Single, UD() As Integer, CC() As Integer) As Integer

Dim I, index As Integer
Dim MinDeltaDensity As Single
Dim DeltaDensity() As Single
Dim InputArr() As Single
ReDim DeltaDensity(1 To 8) As Single
ReDim InputArr(1 To 8) As Single

Dim InputG() As Integer
ReDim InputG(1 To 8) As Integer

Dim InputC() As Integer
ReDim InputC(1 To 8) As Integer

    InputG(1) = UD(X - 1, Y - 1)
    InputG(2) = UD(X, Y - 1)
    InputG(3) = UD(X + 1, Y - 1)
    InputG(4) = UD(X - 1, Y)
    InputG(5) = UD(X + 1, Y)
    InputG(6) = UD(X - 1, Y + 1)
    InputG(7) = UD(X, Y + 1)
    InputG(8) = UD(X + 1, Y + 1)

    InputC(1) = CC(X - 1, Y - 1)
    InputC(2) = CC(X, Y - 1)
    InputC(3) = CC(X + 1, Y - 1)
    InputC(4) = CC(X - 1, Y)
    InputC(5) = CC(X + 1, Y)
    InputC(6) = CC(X - 1, Y + 1)
    InputC(7) = CC(X, Y + 1)
    InputC(8) = CC(X + 1, Y + 1)

    InputArr(1) = RadArr(X - 1, Y - 1)
    InputArr(2) = RadArr(X, Y - 1)
    InputArr(3) = RadArr(X + 1, Y - 1)
    InputArr(4) = RadArr(X - 1, Y)
    InputArr(5) = RadArr(X + 1, Y)
    InputArr(6) = RadArr(X - 1, Y + 1)
    InputArr(7) = RadArr(X, Y + 1)
    InputArr(8) = RadArr(X + 1, Y + 1)

    For I = 1 To 8
        DeltaDensity(I) = Abs(InputArr(I) - Voxel)
    Next I
```

```
'   Search for the minimun distance between RadArr(k,j) and its surrounded 8 voxels -
-
    MinDeltaDensity = 100#
    index = 9
    For I = 1 To 8
       If InputG(I) <> 1 And InputC(I) <> 0 Then
           If DeltaDensity(I) <= MinDeltaDensity Then
               MinDeltaDensity = DeltaDensity(I)
               index = I
           End If
        End If
    Next I
    MinDeltaDensityIndex = index

 'For I = 1 To 8
 '       If InputG(I) = 1 Then
 '           DeltaDensity(I) = 100#
 '       End If
 '   Next I
'
'    MinDeltaDensity = DeltaDensity(1)
'    index = 1
'    For I = 2 To 8
'        'If DeltaDensity(I) <> 100# Then
'            If DeltaDensity(I) <= MinDeltaDensity Then
'                MinDeltaDensity = DeltaDensity(I)
'                index = I
'            End If
'        'End If
'    Next I
'
'    If MinDeltaDensity = 100# Then
'        MinDeltaDensityIndex = 9
'    Else
'        MinDeltaDensityIndex = index
'    End If

End Function
```

MinimumOf2dArrayMu

Qualifiers: Public
Returns: Variant

```
Function MinimumOf2dArrayMu()
    Dim I As Integer, j As Integer
    Dim Minimum

    Minimum = Mu(1, 1)
    For I = 1 To NV
        For j = 1 To NV
            If Mu(I, j) < Minimum Then
                Minimum = Mu(I, j)
            End If
        Next j
    Next I

    MinimumOf2dArrayMu = Minimum
End Function
```

FILEMSG.BAS

Mod Date	Wed Nov 16 12:51:40 1994
Size	2213

Subroutines

FileMsg

Qualifiers:	Public		
Arguments:	FileName	String	By Ref.
		Variant	By Ref.
	Section	Integer	By Ref.

```
Sub FileMsg (FileName$, Section%)
    'Routine to display help message on screen
    'The help file name and the section number is
    'passed as arguments

    'Global variables used
    '   None

    'Determine path for message file
    MsgFile$ = App.Path + "\" + FileName$

    'Be sure file exists
    Fil$ = Dir$(MsgFile$)
    If Fil$ = "" Then
        Msg$ = "File " + MsgFile$ + " not found"
        MsgBox Msg$, 48, "FILEMSG"
        Exit Sub
    End If

    'Create newline string
    NL$ = Chr$(13) + Chr$(10)

    'Open message file for reading
    NumFile% = FreeFile
    Open MsgFile$ For Input As #NumFile%

    'Find specified section
    Do Until EOF(NumFile%)
        Line Input #NumFile%, FileTxt$
            If Left$(FileTxt$, 1) = ">" Then
            If Val(Mid$(FileTxt$, 2)) = Section Then
                Exit Do
            End If
        End If
    Loop

    'Did we reach end of file during search?
    If EOF(NumFile%) Then
        Msg$ = "Message section" + Str$(Section) + " not found"
        MsgBox Msg$
        Exit Sub
    End If

    'Extract message box type and title
    FileTxt$ = RTrim$(LTrim$(Mid$(FileTxt$, 2)))
    FileTxt$ = Mid$(FileTxt$, InStr(FileTxt$, ",") + 1)
    TypeNum% = Val(FileTxt$)
    Title$ = LTrim$(Mid$(FileTxt$, InStr(FileTxt$, ",") + 1))
```

```
    'Loop through all sections of block
    Do

        'Clear message string
        Msg$ = ""

        'Read message section
        Do Until EOF(NumFile%)
            Line Input #NumFile%, FileTxt$
            If Left$(FileTxt$, 1) = ">" Then
                Exit Do
            End If
            Msg$ = Msg$ + FileTxt$ + NL$
        Loop

        'Chop off any ending blank lines
        Do While Right$(Msg$, 4) = NL$ + NL$
            Msg$ = Left$(Msg$, Len(Msg$) - 2)
        Loop

        'Display message block
        If Msg$ <> "" Then
            MsgBox Msg$, TypeNum%, Title$
        End If

    'Continue block if > was by itself
    Loop While LTrim$(RTrim$(FileTxt$)) = ">"

    'We've finished with file
    Close NumFile%
End Sub
```

GLOBAL.BAS

Mod Date Fri Oct 04 12:45:13 1996
Size 2308

Declarations

```
Attribute VB_Name = "GLOBAL"
Option Explicit
Global WorkingDirectory As String
Global Const AspectRatio = (8 / 7)
' Clipboard formats

Global Const CF_LINK = &HBF00
Global Const CF_TEXT = 1
Global Const CF_BITMAP = 2
Global Const CF_METAFILE = 3
Global Const CF_DIB = 8
Global Const CF_PALETTE = 9
' Button parameter masks

Global Const LEFT_BUTTON = 1
Global Const RIGHT_BUTTON = 2
Global Const MIDDLE_BUTTON = 4
' Show parameters

Global Const MODAL = 1
Global Const MODELESS = 0
Global Const NumberOfColors = 256
Global Const SizeOfBMFH = 14
Global Const SizeOfBMIH = 40
Global Const SizeOfRGBQ = 4
Global Const SizeOfColorTable = (NumberOfColors * SizeOfRGBQ)
' MousePointer

Global Const DEFAULT = 0          ' 0 - Default
Global Const ARROW = 1            ' 1 - Arrow
Global Const CROSSHAIR = 2        ' 2 - Cross
Global Const IBEAM = 3            ' 3 - I-Beam
Global Const ICON_POINTER = 4     ' 4 - Icon
Global Const SIZE_POINTER = 5     ' 5 - Size
Global Const SIZE_NE_SW = 6       ' 6 - Size NE SW
Global Const SIZE_N_S = 7         ' 7 - Size N S
Global Const SIZE_NW_SE = 8       ' 8 - Size NW SE
Global Const SIZE_W_E = 9         ' 9 - Size W E
Global Const UP_ARROW = 10        ' 10 - Up Arrow
Global Const HOURGLASS = 11       ' 11 - Hourglass
Global Const NO_DROP = 12         ' 12 - No drop
'Definitions corresponding to BMP files

Type BITMAPFILEHEADER
    bfType      As Integer
    bfSize      As Long
    bfReserved1 As Integer
    bfReserved2 As Integer
    bfOffBits   As Long
End Type
Type BITMAPINFOHEADER
    biSize          As Long
    biWidth         As Long
    biHeight        As Long
    biPlanes        As Integer
    biBitCount      As Integer
    biCompression   As Long
    biSizeImage     As Long
    biXPelsPerMeter As Long
    biYPelsPerMeter As Long
```

```
    biClrUsed       As Long
    biClrImportant  As Long
End Type
Type RGBQUAD
    rgbBlue     As String * 1
    rgbGreen    As String * 1
    rgbRed      As String * 1
    rgbReserved As String * 1
End Type
'6-28-95 hsieh added
'define max and min values of density used for
'color palette calculateion from keystrokes

Global MuRealMin As Single
Global MuRealMax As Single
Global ImgMax As Single
Global ImgMin As Single
Global SelfDefineIndex As Integer
```

VITA

Hsuan-Tsung Hsieh

Candidate for the Degree of

Doctor of Philosophy

Dissertation: QUANTIFICATION OF POROUS MEDIA USING GAMMA RAY TOMOGRAPHY

Major Field: Environmental Science

Biographical:

Education: Graduated from Ming-Dao High School, Taichung, Taiwan, R. O. C. in May 1982; received Bachelor of Science degree in Geology from National Taiwan University, Taipei, Taiwan, R. O. C. in May 1986; received Master of Science degree in Geology from University of Rochester, New York in August 1992; completed the requirements for the Doctor of Philosophy degree with a major in Environmental Science at Oklahoma State University in December 1997.

Experience: Employed by the Geology Department at National Taiwan University in Taipei, Taiwan as a research associate, 1988-89; Employed by the Geology Department at University of Rochester in Rochester, New York as a research assistant, 1990-92; Employed by the Biosystems and Agricultural Engineering Department at Oklahoma State University in Stillwater, Oklahoma as a research assistant, 1993 to present.

Professional Memberships: American Geophysical Union, American Society of Agricultural Engineering.