IN-VIVO ASSESSMENT OF TRABECULAR BONE STRUCTURE AT THE DISTAL RADIUS

By

CHRISTOPHER LANE GORDON, M.Sc.

A Thesis

Submitted to the School of Graduate Studies

in Partial Fulfilment of the Requirement

for the Degree

Doctor of Philosophy

McMaster University

January 1997

DOCTOR OF PHILOSOPHY (1997)
(Medical Physics)

McMaster University
Hamilton, Ontario

TITLE: In-vivo Assessment of Trabecular Bone Structure at the Distal Radius.

AUTHOR: Christopher Gordon, B.Sc.,M.Sc. (McMaster University)

SUPERVISOR: Dr. C.E. Webber

NUMBER OF PAGES: xvi, 279

ABSTRACT

Loss of bone mass has long been recognized as a major factor which makes bones brittle and susceptible to fracture. Currently bone mass is measured using a dual energy photon transmission technique, and a fracture risk is derived from comparison with reference normal values. Although the risk of fracture increases as bone mass decreases, variations in trabecular bone architecture can also affect strength. Consequently, trabecular bone architecture is often cited as a factor which might contribute significantly to fracture risk. Currently, estimates of trabecular bone structure are derived from biopsy studies. Such studies are invasive, destructive, cannot be used routinely in patients or volunteers, and certainly cannot be repeated at the same site to obtain longitudinal measurements. If routine clinical assessments of architecture are to be made, it is necessary to determine which imaging modality best reveals structure in a non-invasive manner. It is also necessary to determine how the competence of the structure can best be expressed quantitatively.

This work has examined ways of assessing trabecular bone structure at the distal radius in-vivo to better understand the contribution of architecture to fracture risk. To this end, it proceeded on four major fronts. First, images of sufficient resolution were acquired using a commercial pQCT scanner and a clinical MR imager. Second, the image processing software necessary to segment the imaged trabecular structure was developed. Third, two indices were proposed to quantify the connectivity of the

segmented structure. One index was derived from the application of trabecular strut analysis to a skeletonized representation of the bone network. The other quantified the marrow space by deriving a mean hole area and maximum hole area of the bone structure as it appears in two dimensions. The clinical value of these indices was tested by conducting pilot studies which examined the ability of the indices to discriminate a small group of Colles fracture patients from the normal population and to reflect normal age related changes in structure. The proposed structural parameters better discriminated Colles' fracture patients than did measures of bone mineral density.

The fourth and last stage of this work examined the proportion of the variance in compressive strength of a group of radius bones that can be accounted for by bone mineral density and bone architecture. In seeking the features that were the most reliable indicators of bone strength, a combination of the mean hole area and maximum hole area had the highest correlation with peak load at fracture. This held true whether these two variables were derived from pQCT or MR images. Therefore, these structural indices may represent a potentially exciting and promising means of discriminating fracture outcomes and monitoring changes in trabecular bone structure.

ACKNOWLEDGEMENT

I would like to thank my supervisor Dr. Colin Webber for his suggestions, guidance and endless patience which proved to be necessary ingredients for this work. I also express thanks to Dr. C. Nahmias and Dr D. Chettle for their valuable input.

I am also indebted to Nicholas Christoforou and Leslie Beaumont. Nicholas did not hesitate to devote many of his weekends during the time it took for us to develop a scanning protocol. Leslie was the source of many hours of conversation and advice which helped to lighten many days.

I must also say a special thank-you to my wife, Sharon, and my daughter, Rebecca. Sharon worked tirelessly to provide me with the free evenings needed to complete this project. Rebecca continues to provide me with many hours of wonder and amusement.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.0 Introduction

Fractures associated with osteoporosis pose a substantial threat to the health of elderly individuals. Hip fractures, for example, are a significant cause of morbidity and disability in older women(Jensen and Tondevald 1979). The restriction of activity which follows a fracture often leads to loss of self-confidence which may further develop to psychologic stress (Thomas et al 1974). To improve the quality of life in these individuals afflicted by osteoporosis, it is estimated that 7-10 billion dollars a year are spent in North America on medical and nursing care. It is generally felt that this enormous health care expenditure would be significantly reduced if an early diagnosis of osteoporosis could be made so that the appropriate preventative therapeutic interventions could be instituted.

A reduction in bone mass is included in most definitions of osteoporosis. Therefore, techniques which have the capacity to measure bone mass with a high degree of accuracy and precision should be able to discriminate between individuals with osteoporosis and the normal population. This implies that identification and assessment of persons with the disease can be improved through technical innovations. This is not the case. Considerable advances in measurement techniques have improved the ability of clinicians to evaluate the bone mineral status of the entire skeleton or at specific sites. Even with these significant

improvements in accuracy and precision, the risk of fracture for a given individual cannot be uniquely determined from bone mass measurements. This inability of bone mass to predict future fractures in an individual implies that there are other factors which contribute to a fracture outcome.

One factor which is thought to contribute to fracture risk is trabecular bone architecture. To better understand the contribution of architecture to fracture risk, methods of quantifying architectural changes are required. To this end, this work examines ways to advance measurements of trabecular architecture into the clinical arena by providing a means of quantifying trabecular bone structure in-vivo.

1.1 Basic Physiology of Bone

Body motion results as the bones in the skeleton provide the support and leverage necessary to transmit the various axial, rotary, and transaxial forces generated by our muscles. The skeletal system itself is comprised of compact cortical bone and the less dense trabecular bone. Whether cortical or trabecular, bone tissue can be further divided into two phases: an organic collagen matrix and a mineral phase; mainly hydroxyapatite. These two phases of bone undergo continuous turnover throughout life. This turnover is necessary to permit repair of microdamage in bone and provides a mechanism for the release of calcium into the circulatory system to satisfy the body's demands. The balance that exists between the process of bone formation and the process of

bone resorption determines the rate of change of bone mass. During the years marked by childhood and adolescent growth the balance between the formation and resorption process favours formation. However, in the normal aging process of the adult female skeleton the balance is shifted in favour of the resorption process such that approximately 1% of the bone mineral mass is lost annually. Most of this bone mineral loss occurs from the trabecular bone compartment which has a turnover rate 3 to 8 times faster than that in compact bone. With a greater rate of mineral loss from trabecular bone, the most frequent fracture sites are those with greater proportions of trabecular bone, namely the proximal femur, vertebrae, and the distal end of the radius.

Osteoporosis is defined as a reduction of the amount of mineral (hydoxyapatite) in bone such that the bone has an increased risk of fracture. Biochemically, osteoporotic bone is not different from normal bone (Wasserman and Burzel 1987). The differences and subsequent risk of fracture between normal bone and osteoporotic bone arise partly from the amount of bone mineral present and the structural arrangement of the mineral. The extent to which this difference in mineral mass can be used to discriminate osteoporotic subjects from the normal population is examined in the next two sections.

1.2 Bone Mass Measurements and Fracture Risk.

The physical strength of bone is directly related to its mineral content (Chalmers and Weaver 1966, Arnold 1973). This

suggests that measurements of the bone mineral content at sites most susceptible to fracture should be the most accurate method of assessing the risk of future fracture. The technology for noninvasive measurements of bone mass has improved considerably. Two techniques most commonly used to measure bone mass are Dual Energy X-ray Absorptiometry(DXA), and Quantitative Computed Tomography (QCT). DXA allows for precise assessments of bone mineral content at peripheral and axial sites at a low startup and operating cost (Gluer et al 1990, Orwall and Oviatt 1991). At the vertebrae DXA measurements may be performed posterior-anteriorly or laterally. The major limitation of DXA is that it cannot distinguish between cortical bone and trabecular bone. An integral bone mass or density is measured. This limitation is offset somewhat at the lumbar spine by performing lateral measurements. A lateral examination of the lumbar spine allows an almost exclusive measurement of trabecular bone because only the vertebral body is measured. Despite this inability to separate trabecular bone from cortical bone, DXA reveals differences between mean values for a population with established osteoporosis and the normal population although ranges may overlap (Pouilles et al 1991, Nuti and Martin 1992, Overgaard et al 1992). At the spine these differences are increased somewhat for lateral examinations in comparison to a posterior-anterior examination (Guglielmi et al 1994, Mazess et al 1995). Most importantly, DXA measurements have shown that fracture rates in some study populations may decrease in response to therapies such as estrogen (Weiss et al 1980, Kiel et al 1987).

QCT assesses changes in trabecular bone mass at the lumbar spine (Cann et al 1985). Because the metabolic rate in vertebral trabecular bone is substantially greater than the cortical bone which surrounds it, the ability to measure solely trabecular bone is a significant advantage over the integral density recorded by DXA. Widespread application of QCT measurements has been limited by the high cost and limited accessibility to clinical CT scanners. Despite its limited use, a measurement of spinal trabecular bone density has been shown to be diagnostically superior to DXA for identifying persons at increased risk of vertebral fracture (Reinbold et al 1986, Guglielmi et al 1994, Laval-Jeantet et al 1995). This improved fracture discrimination with QCT results because the bone mass reduction detected is significantly higher than that observed by DXA (Genant et al 1987, Guglielmi et al 1994).

Based on a measurement of bone mass or density, the identification of patients at a high risk for osteoporotic fracture involves the use of statistical methods. Clinically, the most commonly used statistics are the Z-score and the T-score. The Z-score for a patient is defined as the magnitude of the deviation from the mean result for a group of aged-matched controls divided by the standard deviation associated with the mean calculated for the age-matched control group. The T-score is defined in a similar fashion except that the patient measurement is compared to a young adult reference population. A T score of -2.5 has evolved as the clinically accepted indicator of osteoporosis with or without the

presence of a fracture. This can be problematic, however. There can exist those women with very low bone mass who are fracture free. For example, it has been shown that for a 60 year old woman with a bone mass that is greater than two standard deviations below the mean, the annual risk of not fracturing a bone is approximately 93% (Hui et al 1989).

1.3 Failure of Bone Mass to Predict Fractures

There is no doubt that the risk of a fracture increases as bone mass and density decrease. However, there is a large overlap between bone mass measurements in normal, nonfractured women and in those who have or will develop fractures (Cann et al 1985, Riggs et al 1990). To illustrate this, figure 1.1 shows a plot taken from the work of Riggs. As noted, the points plotted represent 84 women with one or more nontraumatic vertebral fracture. Over half of these women fall within the 90% confidence limits indicating normal bone density. Reviews of case-control studies which have used a bone mass measurement as the sole determinant of increased fracture risk have all concluded that differences in bone mass between patients with fractures and controls are small and the ranges usually overlap (Mazess 1981, Cummings 1985, Ott 1993). This overlap between bone mass measurements in normal, non-fractured individuals and in those who have fractures is an indication that bone failure is a complex disorder that cannot be predicted by measuring bone mass alone.

The resistance of a bone to fracture is also dependent on

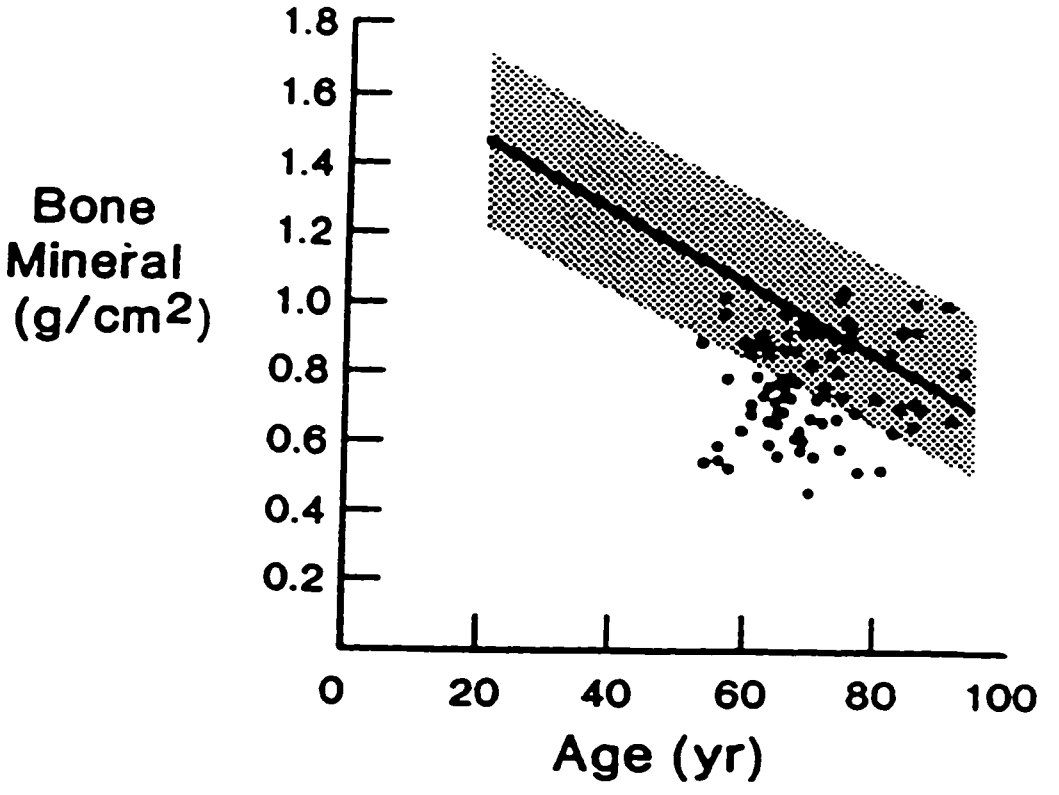


Figure 1.1: Individual lumbar spine BMD values in 84 women with one or more nontraumatic vertebral fractures. The shaded area represents 90% confidence limits, and the line denotes age regression for normal women.

other factors. These factors are illustrated in figure 1.2 and can be categorized as being either inherent to bone or as external factors. As shown, bone geometry, bone quality, the presence of microdamage, and the protective response, following loss of balance can all influence whether or not a bone fractures. Measurements of architectural parameters which are inherent to bone are often cited as most likely to improve a prediction of fracture risk beyond that offered by bone mineral content. This is not surprising because the mechanical behaviour of trabecular bone is partly determined by its degree of anisotropy and connectivity (Townsend et al 1975, Goldstein 1987). Morphological parameters such as trabecular thickness and number density may also indicate strength. For example, by examining the role of trabecular thickness and density in the pathogenesis of vertebral fracture, Kleerekoper et al (1985) concluded that the biomechanical competence of cancellous bone is not only dependent on the absolute amount of bone present but also on the trabecular microstructure. Similarly, Jensen et al (1990) have noted that a considerable change in the mechanical behaviour of trabecular bone can occur when the bone mineral is slightly redistributed so that trabecular bone volume and mass remain unchanged. It appears then, that methods which accurately quantify trabecular bone architecture will improve the identification of those subjects with increased risk of bone fracture.

1.4 The Architecture of Normal and Osteoporotic Bone.

The normal structure of cancellous bone may differ in detail

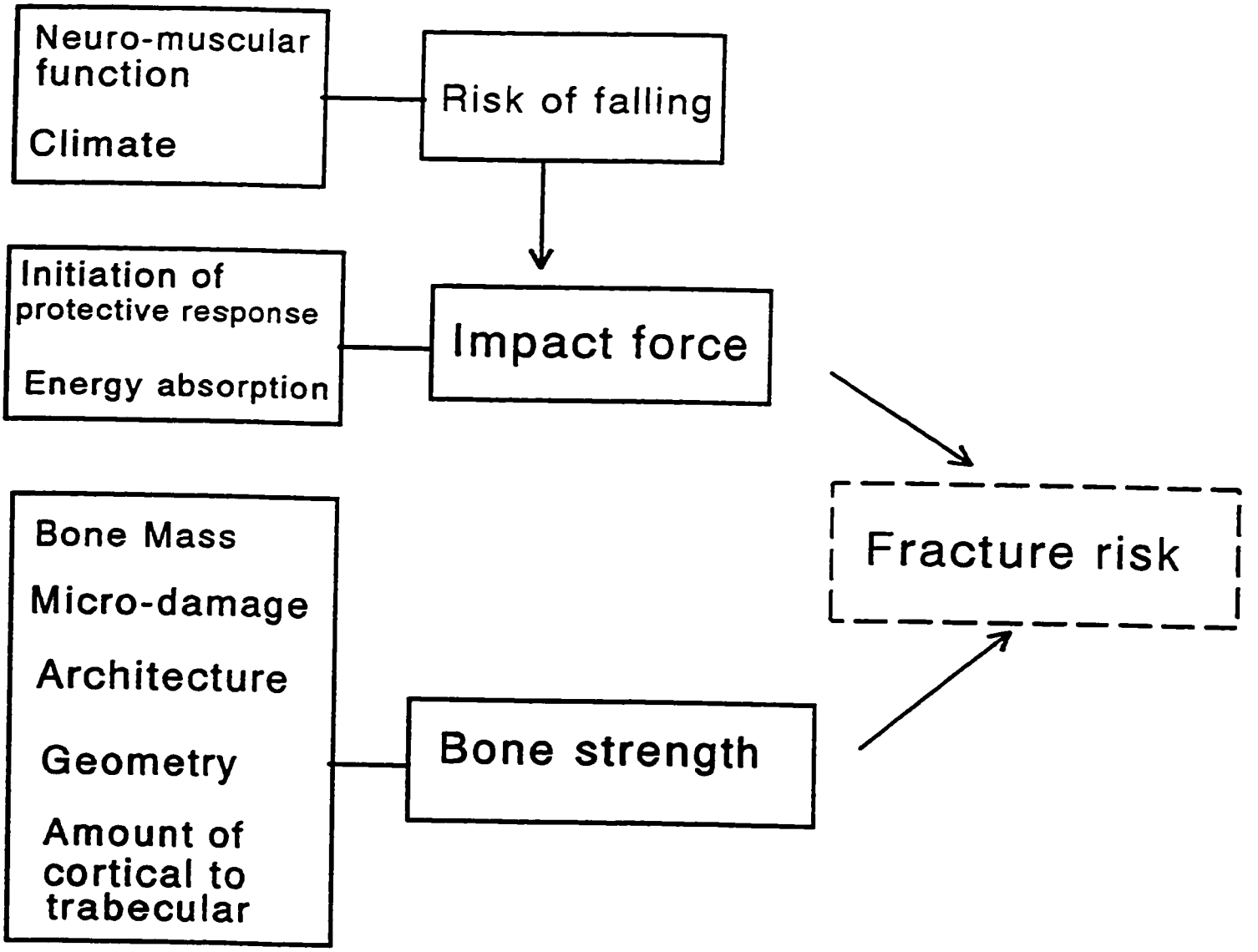


Figure 1.2: Contributing factors which play a role in the determination of fracture risk in a given individual.

and amount at different body sites but consists of the following components. Generally there are combinations of curved plates in which there are holes of various sizes and a meshwork of rods of a variety of lengths (Whitehouse 1977). This description of cancellous bone is sketched in figure 1.3a. The average plate thickness is approximately 0.13 mm while the average size of the holes within each plate is 0.75 mm (Amstutz and Sissons 1969). Similarly, the rods of trabeculae that interconnect the various plates have an average width of 0.16 mm but can range between 0.05 mm to 0.2 mm depending on the body site (Whitehouse 1977).

Osteoporosis is a skeletal condition characterized by a reduction in the mass of bone mineral that disrupts the trabecular bone structure. It is now clear that the loss of cancellous bone mass and accompanying disruption in structure with age occurs principally by a process that removes entire trabeculae rather than a generalized uniform thinning of the whole structure (Parfitt et al 1983; Birkenhager-Frenkl et al 1988). This change is illustrated in figure 1.3b. As shown, those trabeculae that remain are more widely separated and thus less likely to withstand a compressive force. However, there is perhaps a protective response initiated by the cancellous bone structure to maintain mechanical integrity. Vesterby et al (1989a) found that normal postmenopausal women increase iliac crest trabecular thickness with age as a result of the increased load that individual trabeculae have to bear.

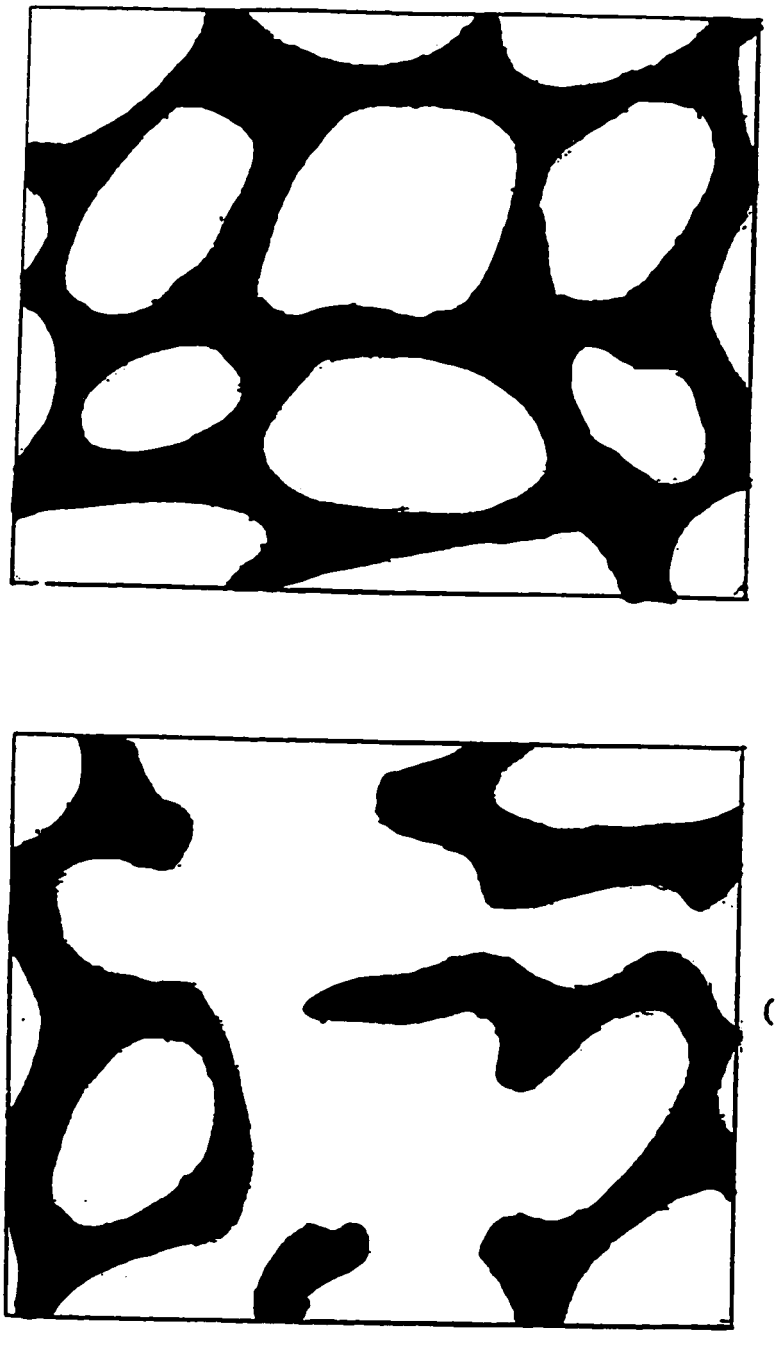


Figure 1.3: A comparison between the appearance of normal trabecular bone (a) and osteoporotic bone (b).

1.5 In-Vitro Parameters to Quantify Bone Structure

The biomechanical competence of bone can only partly be derived from bone mass measurements. With the importance of trabecular bone architecture being demonstrated, research questions have now focused on a measure of bone mass in combination with an assessment of architecture as an indicator of fracture risk. Non-invasive measures of bone mass have become extremely accurate and precise. This is not the case, however, for measurements of trabecular bone architecture. The challenge for bone researchers over the last decade has been to develop indices that accurately quantify the three dimensional structure of cancellous bone. Solutions to this challenge have been derived in two ways. First, in-vitro structural assessments can be made from two-dimensional (2D) histologic sections prepared from biopsy. Such an approach has one obvious limitation. The three dimensional structure of trabecular bone cannot be fully understood from isolated 2D sections. Second, the limitation of 2D analysis can be overcome by the direct examination of three-dimensional (3D) bone structure in-vitro with the use of high-resolution computed tomography. Larger samples are required to fully represent the 3D structure and so samples obtained from autopsy are often analyzed. Both approaches to in-vitro structure assessment have yielded indices which reflect changes in bone structure as a result of aging, disease, and response to therapy. These indices are examined in the next two sections.

1.5.1 Parameters derived from histological sections

A number of interesting parameters have been developed to quantify trabecular bone structure from micro-sections. The first and simplest was the direct measurement or calculation of trabecular width, separation, and number density from a biopsy sample(Parfitt et al 1983). The biopsy sample to be measured is prepared, magnified and a calibration grid is superimposed. With the aid of standard image analysis packages, the area (A_b) and perimeter (P_b) of cancellous bone is determined. Given that the analysis procedure is calibrated to an external bone standard and that the total area of the overlying grid is delineated by A_t, then the following quantities of bone architecture can be derived:

$$(MTPT) = C\left(\frac{A_b}{P_b}\right) \qquad 1.1a$$

$$(MTPS) = C\left(\frac{(A_t - A_b)}{P_b}\right) \qquad 1.1b$$

$$(TBV) = \frac{A_b}{A_t} \qquad 1.1c$$

$$(MTPD) = C\left(\frac{P_b}{A_t}\right) \qquad 1.1d$$

where MTPT is the mean trabecular plate thickness, MTPS is the mean trabecular plate spacing, TBV is the trabecular bone volume, and MTPD is the mean trabecular plate density. Each parameter is listed

as being proportional to two primary measurements, area and perimeter. Each can then be relayed as a three dimensional measure by the insertion of appropriate constants into the equations (Parfitt et al 1983).

To quantify the volume of trabecular bone and of marrow space, the star volume technique has been applied (Vesterby et al 1989b). Star volume is defined as the mean volume of all the parts of an object which can be seen unobscured in all directions from a particular point inside the object to the boundary of the object. An illustration of the star volume technique is given in figure 1.4. It is applied to microsections of bone through the following steps. First, a seed point is picked at random within the marrow space and a selected number of rays are drawn outward and isotropically until intersecting a bone boundary. If the average length of all rays is l_o then the mean star volume V^* is calculated by the following equation.

$$V^* = \frac{\pi}{3}(l_o)^3 \qquad 1.2$$

A scaling constant ($\frac{1}{3}\pi$) is used to relate a two dimensional measurement to a three dimensional index of bone. As illustrated by figure 1.4, when the seed point is picked in marrow and the rays "star" out from the seed point, the algorithm returns a measure of marrow volume. When the seed point is picked inside a trabeculae the algorithm returns a measure of trabecular volume. Consequently, with loss of trabecular bone the marrow star volume increases and the trabecular star volume decreases.

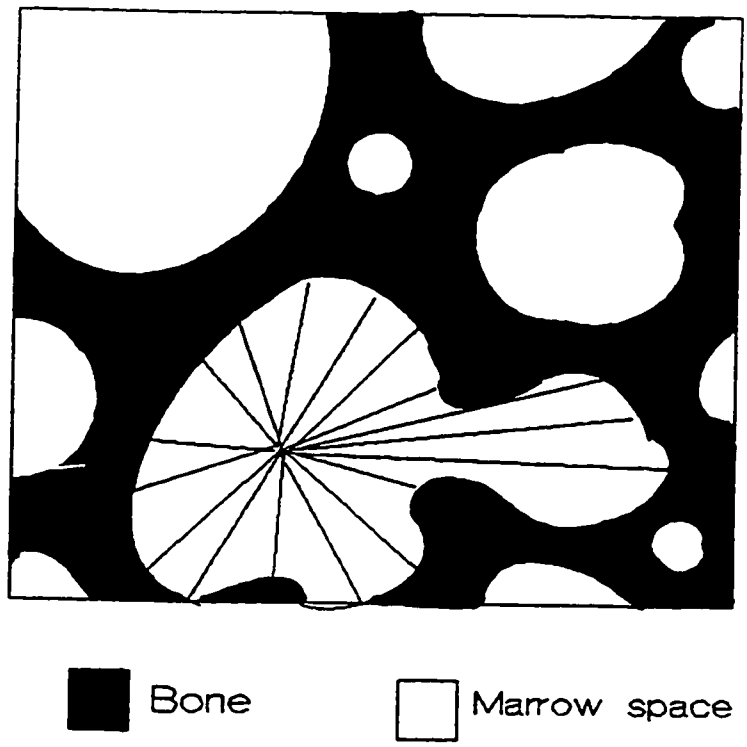


Figure 1.4: A star volume measurement of a marrow pore is indicated. From a seed point rays are drawn - outward and isotropically until intersecting a boundry.

To assess the degree of connectivity between plates and rods, strut analysis has been applied (Parisien et al 1992). In the case of strut analysis the bone section to be examined is magnified and is considered to consist of a number of 1-dimensional struts. This consideration of the bone section is illustrated in figure 1.5. The junction between three struts is defined as a node. A strut that is connected at one end to a node and is free at the other end is labelled a free end. The number of node to node and node to free end struts are counted to determine the connectedness of the bone section. A well connected bone is characterized by a large number of node-node struts and few node-free end ones. A change in connectivity such as a break or removal of a strut results in the number of node-node struts being decreased by one while the number of node-free end struts must increase by two.

Connectivity has also been quantified by a trabecular bone pattern factor (Hahn et al, 1992). The basis of the trabecular bone pattern factor(TBPf) can be described by the relation of convex to concave surfaces present in the bone lattice. With the aid of figure 1.6 calculation of TBPf involves the following steps. First, by means of an automatic image analysis system, the perimeter (P1) and area (A1) of the bone section is calculated. Second the bone image is dilated by adding a layer one pixel thick to the bone contour. This transformation is illustrated in figure 1.6b. The perimeter (P2) and area (A2) are remeasured. The TBPf is then calculated from the following equation.

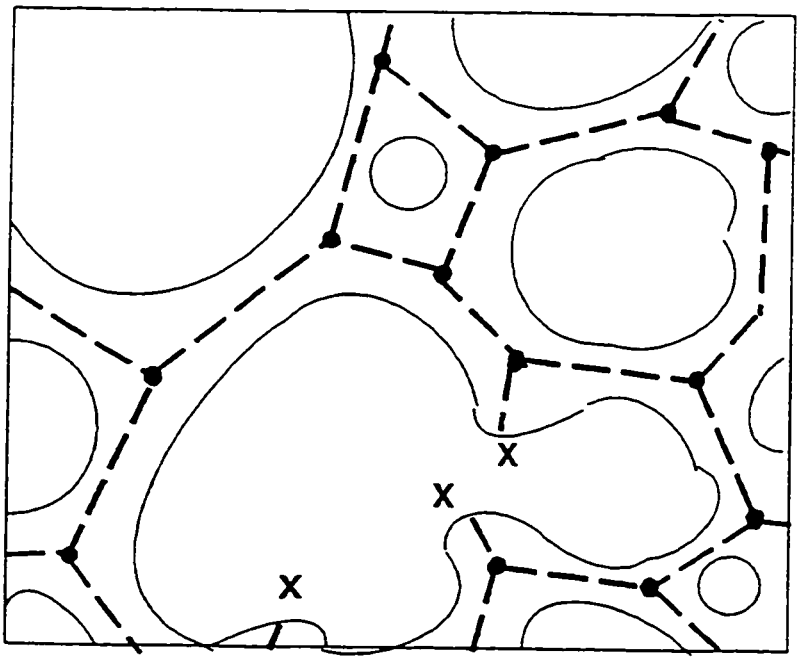


Figure 1.5: In trabecular strut analysis the two-dimensional representation of the bone structure is represented by a series of one-dimensional struts shown here as broken lines. Nodes are indicated by (●) and free ends by (x).

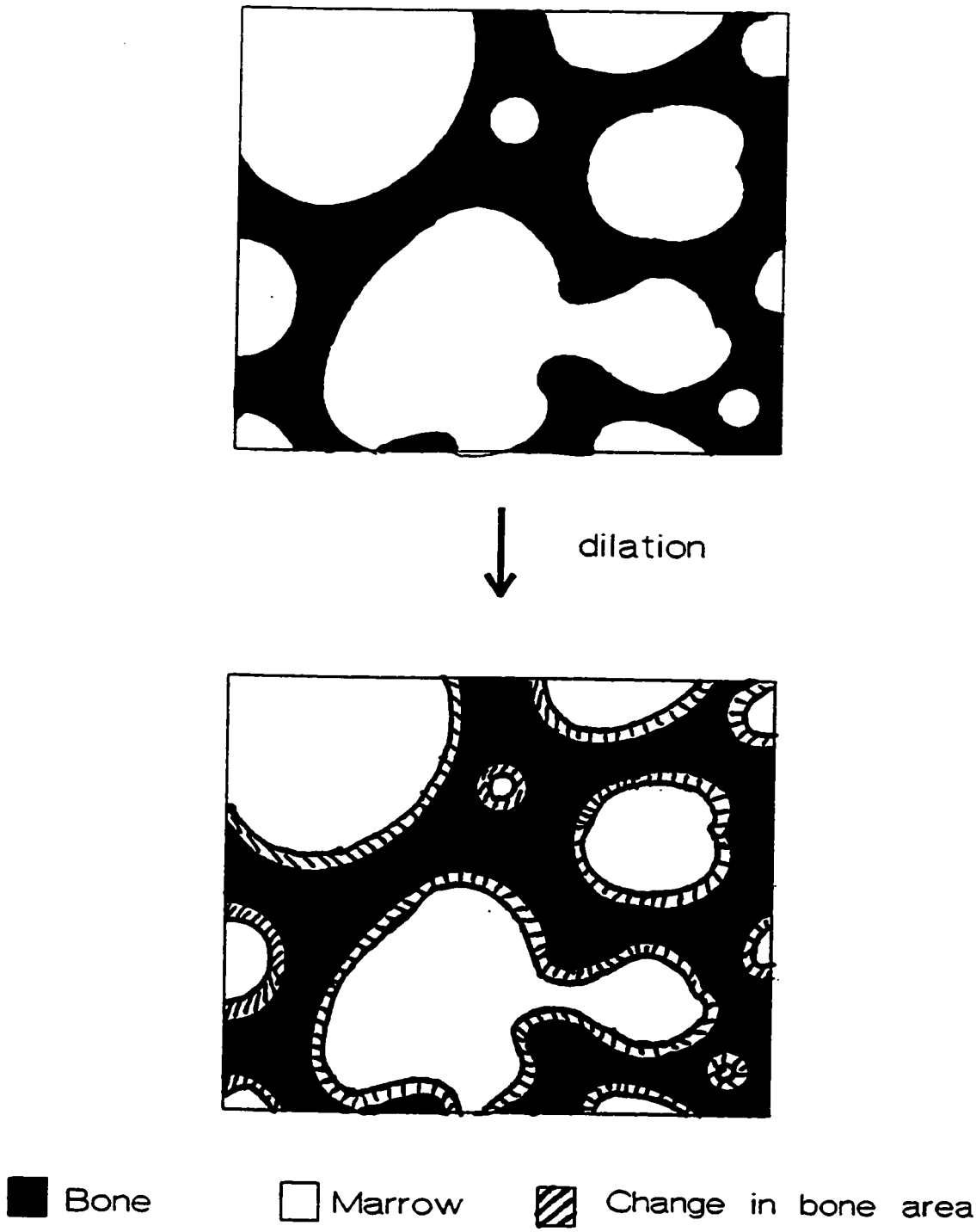


Figure 1.6: A calculation of the trabecular bone pattern factor involves the determination of the bone perimeter before and after dilation. The dilation step helps to differentiate a well connected bone network from one with many breaks.

$$TBPf=\frac{(P1-P2)}{(A1-A2)} \qquad 1.3$$

The dilation results in a change of area and perimeter which indicates the relation of convex to concave bone surfaces. For example, after dilation the bone area always increases (A1<A2) irrespective of the convex or concave nature of the bone. However, the bone perimeter increases for a convex surface (P1<P2) but decreases for a concave surface (P1>P2). Any breaks that occur in the trabecular plate network will result in a decrease in the number of concave surfaces and an increase in those that are convex. Consequently, in the case of a normal well connected bone lattice, the TBPf is low and perhaps negative. For diseased bone with lots of isolated trabeculae TBPf is high.

1.5.2 Parameters derived by diagnostic imaging methods

A direct examination of three-dimensional bone structure in-vitro can be obtained from high-resolution computed tomography images. In the literature this analysis is often referred to as micro-computed tomography to distinguish it from high-resolution computed tomography. The distinction is based on differences in achievable resolution. Micro-CT systems can achieve resolutions less than 100 μm while high-resolution CT systems may reach 500 μm.

The first measure of three-dimensional connectivity determined by micro-computed tomography was obtained by Feldkamp et al(1989). In this application, a combination of a highly focused X-ray source, an image intensifier, and rotation of the bone sample to

obtain the required projections resulted in a resolution of 75 μm. At this resolution the number, orientation, and size of trabeculae in all three dimensions were readily extracted from the images. A number of other applications of micro-CT analysis to the study of trabecular bone structure have since followed. For example, finite element models have been derived from the CT data set to predict the material properties of trabecular bone (Muller and Ruegsegger 1996). Also, attempts have been made to quantify 3D connectivity using Euler characteristics (Odgaard and Gundersen 1993) and fractals (Majumdar et al 1993).

Although the primary use of micro-CT has been to acquire a fully three-dimensional representation of trabecular bone, it has also been used for the non-destructive evaluation of two-dimensional indices of architecture (Ruegsegger, Koller, and Muller 1996). The thin image slices provided by these microtomographic systems allows for a large number of sections of the sample to be analyzed without the often time consuming steps of slicing and mounting which are required during histologic preparations. As a result of this efficiency, microtomographic analysis of trabecular architecture has emerged as a nondestructive biopsy technique to quantify in-vitro structure (Davis and Wong 1996).

1.6 In-vivo Assessment of Trabecular Bone Structure

The indices of cancellous structure previously discussed are all derived in-vitro from samples of bone obtained by biopsy. Such procedures cannot be used routinely in patients or volunteers and

certainly cannot be repeated at the same site to obtain longitudinal measurements. An in-vivo assessment of trabecular bone architecture has the immediate potential for serial measurements and application in everyday clinical situations. The requirement for an in-vivo assessment of trabecular bone architecture is the acquisition of an image of sufficient resolution to allow quantitation. There are two methods which may provide images of sufficient resolution to allow an assessment of architecture in-vivo. They are Quantitative Computed Tomography (QCT) and Magnetic Resonance Imaging (MRI). These two modalities exploit different physical principles to image trabecular bone structure. An in depth examination of these principles is reserved for later chapters. However, the application of each to an assessment of structure is examined briefly in the following sections.

Ultrasound has also been applied to the study of trabecular bone density and structure. Its basic principles as they apply to the study of bone density and architecture are also discussed in the following sections. However, because ultrasound cannot provide an image of sufficient resolution which can be interrogated for structure, it is not considered further beyond this introductory chapter.

1.6.1 Quantitative Computed Tomography (QCT)

Quantitative Computed Tomography (QCT) was developed to determine bone mineral density predominantly at the spine (Cann and Genant 1980). QCT uses the method of tomographic reconstruction of

X-ray profiles to obtain the transaxial distribution of attenuation coefficients in Hounsfield units. Using a thresholding algorithm trabecular bone is isolated from cortical bone. A reference standard containing well defined amounts of bone mineral equivalent material is used to calibrate the Hounsfield units to an equivalent bone mineral density.

The widespread use of QCT as a screening tool for osteoporosis has been limited by access to the CT scanners. However, systems dedicated to bone mineral measurements at peripheral body sites such as the radius are now available commercially. Like their axial counterparts these peripheral QCT (pQCT) systems have been shown to be highly reproducible when applied in serial examinations of trabecular and cortical bone density in a postmenopausal population (Muller et al 1989).

A determination of bone density by QCT at the spine or radius requires the performance of calculations on the image matrix. The success of assessing structure by CT depends strongly on acquiring images of sufficient resolution to visualize the trabecular network. In healthy persons this network consists of trabeculae ranging in thickness from 0.1 mm to 0.4 mm (Whitehouse 1977). These trabeculae inter-connect to produce an average marrow space of 0.75 mm but can range from 0.2 mm to 2 mm (Amstutz and Sissons 1969). In osteoporotic patients trabeculae become thinner and larger marrow spacings are found due to entire trabeculae being removed (Birkenhanger-Frenkel et al 1988, Vesterby et al 1989a). The spatial resolution achieved with current CT scanners may just be

adequate to identify these physiological changes in trabecular architecture in-vivo. A few studies have shown that by processing the resulting CT image, information reflecting the state of trabecular structure at the spine can be obtained and used to improve the separation of healthy subjects from patients with bone disease (Chevalier et al 1992, Mundinger et al 1993). In these applications, the images of the vertebrae were first processed by such techniques as filtering, edge and contrast enhancement, noise exclusion, and thresholding. From the processed image, indices of texture relating to the total number of trabeculae and intertrabecular spaces were calculated. These indices proved valuable in the identification of a subgroup of patients suffering from fractures but with normal bone mass.

The limited access to conventional, all-purpose CT scanners has prompted the development of dedicated peripheral QCT (pQCT) instrumentation specifically for measurements of bone mass in peripheral sites susceptible to fracture such as the radius (Stebler and Ruegsegger 1983, Hangartner and Overton 1982, Hosie and Smith 1986). Like their axial counterparts, the high resolution pQCT image generated during the assessment of bone mass may be useful for estimating structural parameters of trabecular architecture. One study has already demonstrated the feasibility of extracting structural information from these pQCT images (Durand and Ruegsegger 1991) by using run-length analysis. A close relationship between histomorphometric values and run-length parameters was demonstrated from images of the distal radius and

tibia. However, direct interpretation of the run-length parameters as indicators of structure required simulating two and three-dimensional models of the trabecular architecture.

1.6.2 Magnetic Resonance Imaging(MRI)

Two applications of magnetic resonance imaging have emerged as potential tools for investigating cancellous bone structure. The first method is based on the differences in magnetic permeability that exist between bone mineral and bone marrow. This difference produces significant inhomogeneity in the local magnetic field surrounding the trabecular elements. As a result the lifetime of the signal described by the T_2^* parameter from marrow is shortened as the mass of trabecular bone increases. This effect was first demonstrated in vitro by adding varying amounts of bone powder to water (Davis et al 1986). In-vivo measures of the T_2^* of marrow showed an increase from an epiphyseal to a diaphyseal site which was interpreted as arising from increases in the intertrabecular space due to reductions in trabecular density (Ford and Wehrli 1991). T_2^* measured at the lumbar spine in healthy persons is significantly shorter than in patients with osteoporosis (Wehrli et al 1995).

Along with the approach of relating the decay time of the MR signal from marrow to bone density and structure, MR can be used to acquire images of sufficient resolution to visualize the trabecular network directly. The necessary spatial resolution has been achieved on current clinical systems by means of various technical

approaches. For example, a pixel resolution of 117 μm x 117 μm x 400 μm was reported for images of the phalanges (Wong et al 1991). Such resolution can be achieved for smaller object sizes and with the aid of a local gradient coil and modified pulse sequences. A pixel resolution of 156 μm x 156 μm x 700 μm was achieved for cross-sectional images of the wrist using a specialized wrist coil, modified scanning software, and standard imaging gradients (Foo et al 1992, Majumdar et al 1994). Once imaged by MR, a number of different approaches have been used to assess the structure of trabecular bone. For example, gray scale morphometric granulometries (Yidong et al 1993) and fractal analysis techniques (Majumdar et al 1993) have been applied to images of the wrist. More recently, the mean intercept length was used to assess trabecular width as a function of angle and orientation at the radius and calcaneus (Majumdar et al 1994, Majumdar and Genant 1995). These different approaches that characterize structure are dependent upon the quality of the original image from which they are derived.

1.6.3 Ultrasound

Ultrasound measurements have been proposed as an alternative to BMD for evaluating fracture risk. The smaller size, reduced expense of equipment, and absence of ionizing radiation makes ultrasound measurements more attractive than some forms of densitometry.

The basic physics that makes ultrasound appropriate for bone

mineral assessments is as follows. The velocity of sound (v) within a particular medium is dependent on the density (ρ) of that medium. Quantitatively this relation is given by the following equation.

$$v \propto \frac{1}{\sqrt{\rho}} \qquad 1.4$$

The ultrasound wave propagates through the medium by the separation and compression of neighbouring molecules within the medium. Therefore, it is not surprising that the compressibility is another physical characteristic that affects the velocity of sound through the medium. Compressibility (K) indicates the fractional decrease in volume when pressure is applied to the medium. The velocity of sound is also inversely proportional to the square root of the compressibility of the medium. That is:

$$v \propto \frac{1}{\sqrt{K}} \qquad 1.5$$

The elastic properties of the medium are related to its compressibility. It is measured as the bulk modulus (β) and is inversely related to compressibility.

$$\beta = \frac{1}{K} \qquad 1.6$$

Combining compressibility and density into one equation yields the following relationship:

$$v = \frac{1}{\sqrt{K\rho}} \qquad 1.7a$$

or

$$v=\frac{\sqrt{\beta}}{\sqrt{\rho}} \qquad 1.7b$$

The bulk modulus (β) is often referred to as a measure of the "stiffness" of the medium in resisting an applied force. Therefore, as given in equation 1.7b, a measure of the acoustic velocity through a material is influenced by its density and intrinsic stiffness. These two material properties are important in resisting mechanical stress.

Two uses of ultrasound enable a measurement of bone mineral density. The first is the velocity of the sound wave as it travels through the bone. This is recorded as the speed of sound (SOS). The second ultrasound measure is the frequency attenuation of the sound wave as it traverses the bone. The attenuation is recorded for a broad range of frequencies (0.1-1.0 MHz) and so this measure is referred to as broadband ultrasound attenuation (BUA). A measurement of BUA involves sending a broad band ultrasound pulse through the bone and recording how much of a given frequency is absorbed by the bone. This broad frequency spectrum allows measurement of attenuation to occur over a range of frequencies. Subtracting the intensity values transmitted by the bone from a spectrum obtained by transmission through a weakly attenuating reference medium, such as water, provides the net attenuation at each frequency. This net attenuation is calculated at discrete frequencies and a regression line is derived to obtain the attenuation slope in dB/MHz. This slope is the BUA value.

The clinical utility of an ultrasound measurement has been

demonstrated on two fronts. First, ultrasound detects osteoporosis and increased fracture risk with a sensitivity and specificity comparable to that of other bone mass measurement techniques (Heaney et al 1989, McCloskey et al 1990, Argren et al 1991, Ross et al 1995). Second, a number of studies have shown evidence that ultrasound provides clinically relevant information about bone quality in addition to and distinct from bone mass. For example, BUA and SOS can be related to histomorphometric variables such as trabecular plate separation (Hans et al 1993) and trabecular orientation (Nicholson et al 1994). Most importantly, when postmenopausal and osteoporotic subjects are matched for equal BMD an SOS measure distinguishes the two groups (Brandenburger 1993). This suggests that the value of ultrasound may not be in its ability to predict BMD but rather in the assessment of fracture risk and bone quality. Therefore, this low cost non-invasive technique may have widespread utilization in the future.

1.7 Research overview

The work reported in this thesis examines ways of assessing trabecular bone structure at the distal end of the radius in vivo to better understand the contribution of architecture to fracture risk. To this end, the focus is on four major areas. The first examines ways by which the trabecular bone structure at the radius can be imaged with MRI and pQCT. Secondly, the image processing tools necessary for segmenting the imaged bone structure are discussed. Thirdly, indices to quantify the connectivity and

orientation of the segmented structure are proposed. Finally, the clinical value of the proposed indices are examined through compressive testing of a small group of radii in vitro and by discriminating a group of Colles fracture patients from the normal population.

Chapter 2

PRINCIPLES OF COMPUTED TOMOGRAPHY

2.0 Introduction

In standard radiographic imaging a three dimensional structure is collapsed onto a two dimensional image. The signal at each point on the film represents the summation or line integral of the linear attenuation coefficients along the path defined by the x-ray source to the point on the film. This superposition restricts the diagnostic applications of the detected image. For example the use of a plain film exam to characterize trabecular structure is severely limited. Projection blurs the bone structure in the final image. These limitations could be overcome if it were possible to image just thin cross-sections of the anatomy at a time. X-ray based Computed Tomography (CT) represents an effective way to obtain images of thin sections of the body. An overview of the CT process is given in the following sections. It was derived from the review article compiled by Brooks and Di Chiro (1976) and from the consideration of image reconstruction from projections given by Herman (1980).

2.1 Basis of Computed Tomography

The basic principle behind CT is that the internal structure of an object can be reconstructed from multiple one dimensional projections of the object each obtained at a different angle around the object. Each linear projection, p, consists of a set of ordered numbers each representing the fractions of x-rays transmitted along a line path normal to the direction of the

projection. This line path is referred to as a ray, or ray sum, and quantifies the degree to which the x-ray beam is attenuated by the tissues along its path. This scheme is illustrated in figure 2.1. One simple way by which each of the one dimensional projections can be obtained is shown in figure 2.2. As shown an x-ray source and detector are translated in tandem to obtain a projection. The source and detector assembly are then rotated to obtain another projection at a different angle around the object.

If the cross-sectional layer is divided into a matrix of pixels with dimension Δx then mathematically the fraction of x-rays transmitted along a ray can be written as

$$Ln\left(\frac{I}{I_o}\right) = -(\mu_1+\mu_2+\mu_3+\ldots)\Delta x \qquad 2.1$$

I_o represents the intensity of the x-ray beam before it passes into the cross-section and I gives the beam intensity as it exits the body tissue. The μ's are the linear attenuation coefficients of the successive voxels with thickness Δx. If the beam were monoenergetic each voxel would have a uniquely assigned attenuation coefficient which is determined by the composition of the tissue occupying it and depending on the energy of the beam. In practice the x-ray beam is made up of photons of many energies. Therefore, as it passes through the absorbing tissue, the mean energy of the beam increases as the lower energy photons are filtered out. As the beam is hardened, the attenuation at a point within the cross-section will vary with the quality of the beam passing through it. Therefore, for a spectrum made up of many energies, the linear attenuation coefficient assigned to a voxel will depend on tissue composition as well as on the mean

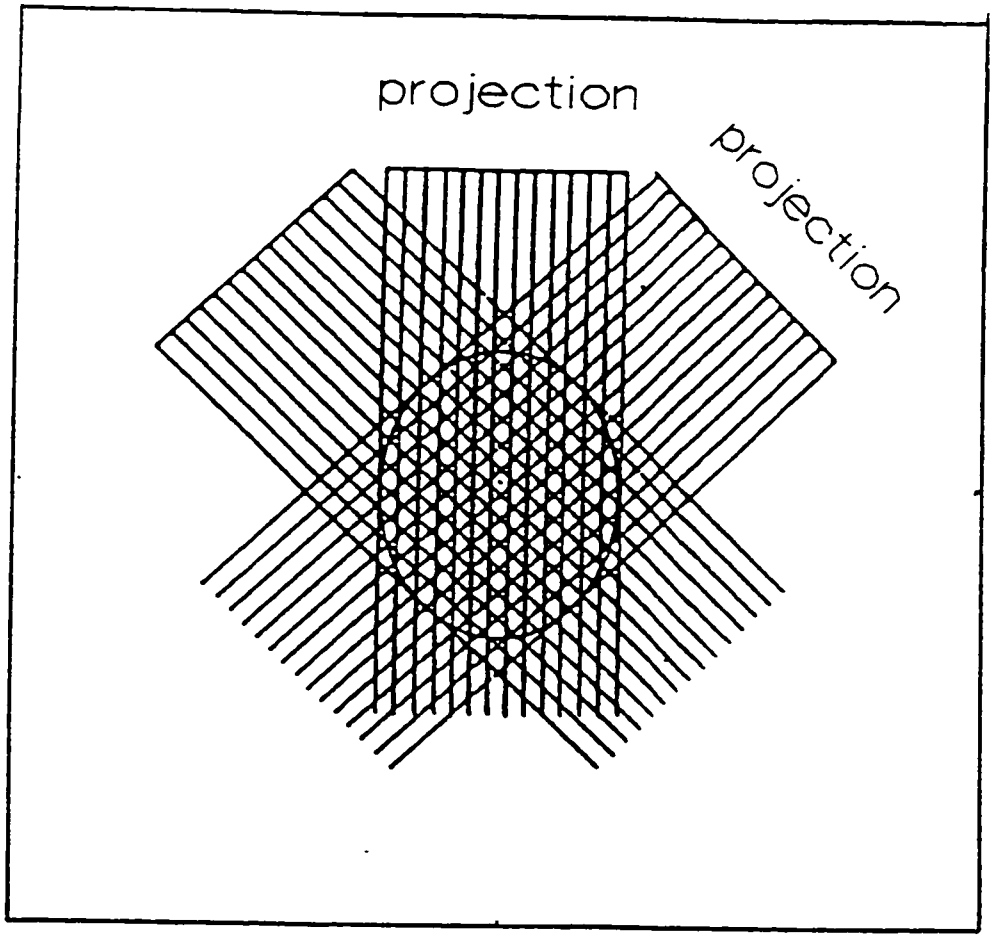


Figure 2.1: A typical scanning pattern used to acquire a set of one dimensional projections from which the internal structure of an object can be reconstructed. The pattern consists of linear translations at successive angular intervals around the object.

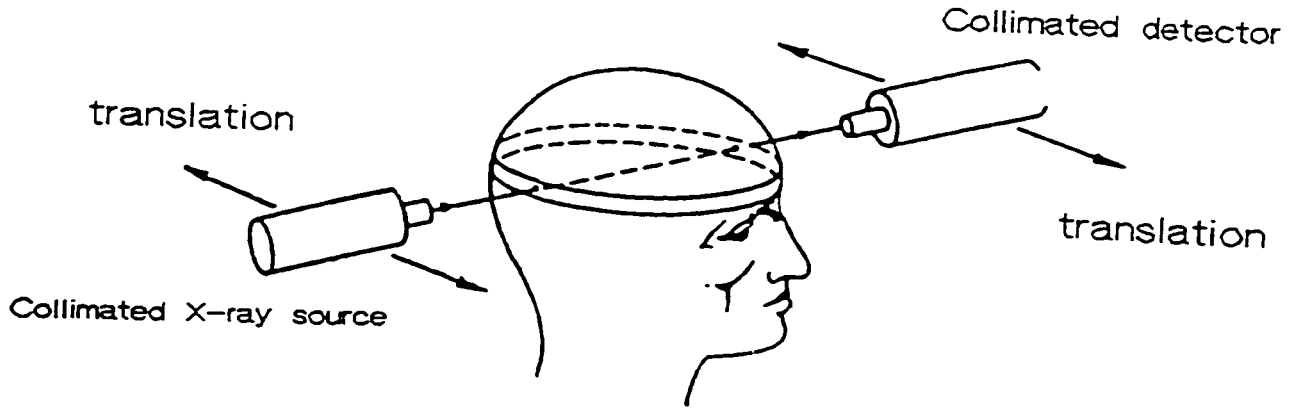


Figure 2.2: By moving an x-ray source in tandem with a collimated detector, transmission measurements can be recorded along selected line paths.

energy of the x-ray beam at the particular location of the voxel in the body cross-section. This beam hardening effect must be corrected for to obtain useful images. Hence, the polychromatic projection data is most often corrected to yield a monochromatic projection data set by means of a second or third order polynomial whose weighting coefficients are determined by the energy profile of the x-ray spectrum and the type of material being imaged (Herman 1980). The essence of CT is to use the corrected transmission data to reconstruct the distribution of the linear attenuation coefficients in the two-dimensional slice.

2.2 Mathematical Description of the Reconstruction Problem

The mathematical exercise is to reconstruct a function f(x,y) which represents the linear attenuation coefficients in the two-dimensional slice. The problem is best approached by defining the co-ordinate systems shown in figure 2.3. Ray paths are described by an (r,s) co-ordinate system which is rotated by the same angle ϕ (with respect to the x-y frame of reference). Therefore, each ray recorded can be specified by an angle ϕ and a distance r from the origin. S gives the path along the ray. In the (r,s) co-ordinate system the projection or ray sum of f(x,y) along a ray is defined by the following integral.

$$p(r,\phi) = \int_{r,\phi} f(x,y)\, ds \qquad 2.2$$

This projection p is proportional to the logarithm of the detector signal. With f(x,y) representing $\mu(x,y)$, then from equation 2.1, p can also be expressed with the following equation based on the detector signal:

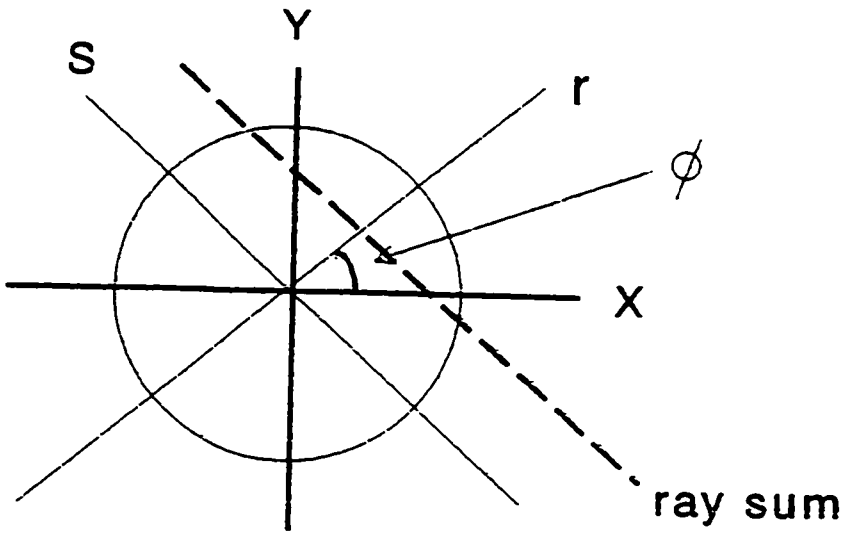


Figure 2.3: A rotated coordinate system used to describe the projection data. Ray sums (dashed line) are specified by their angle ϕ and their distance r from the origin. Distances along the ray sum are defined by s. Points within the object being imaged are described by the fixed (x,y) co-ordinate system.

$$p=-\ln\left(\frac{I}{I_o}\right) \qquad 2.3$$

In theory, f(x,y) is a continuous two-dimensional function and , in principle, can be recovered from an infinite number of projections taken at an infinite number of angles. In practice, $p(r,\phi)$ is measured at a number of discrete positions from which an estimate of f(x,y) can be calculated.

2.3 Analytic Reconstruction: the reconstruction solution

2.3.1 Two-dimensional Fourier reconstruction

The starting point for the analytic reconstruction is to examine the two-dimensional Fourier transform of the density function f(x,y). This is given by

$$F(k_x,k_y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\exp\left[-2\pi i\left(k_x x+k_y y\right)\right]dxdy \qquad 2.4$$

The parameters k_x and k_y are the wave numbers (figure 2.4). The angle of rotation can be defined from the wave numbers and is derived from the following equation.

$$\phi=\tan^{-1}\left(\frac{k_y}{k_x}\right) \qquad 2.5$$

It is also apparent that k_x and k_y can be defined as follows.

$$k_x=k\cos(\phi) \qquad 2.6a$$

$$k_y=k\sin(\phi) \qquad 2.6b$$

Now, from the (r,s) coordinate system defined in figure 2.3, it is clear that

$$r=x\cos(\phi)+y\sin(\phi) \qquad 2.7$$

Combining equations 2.6 and 2.7 leads to the following relation

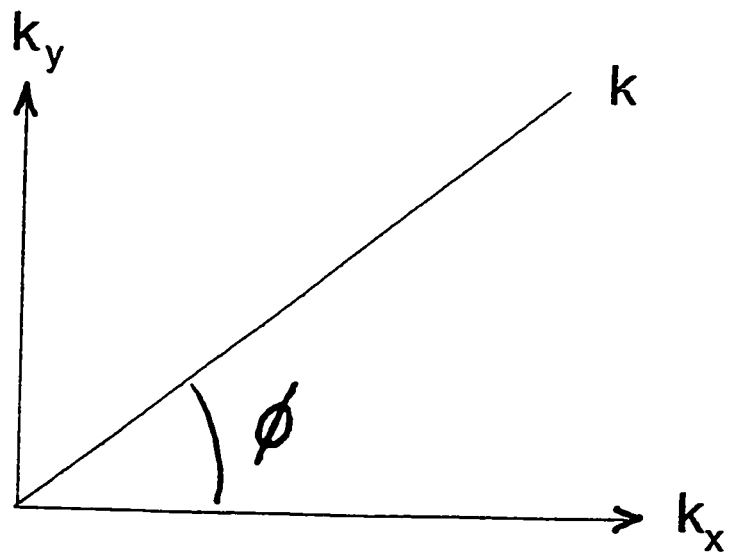


Figure 2.4: Relationship between the orthogonal coordinate system (k_x, k_y) and the rotated coordinate system defined by an angle ϕ relative to the y axis.

$$k_x x + k_y y = kr \qquad 2.8$$

Using this simple geometrical transformation, equation 2.4 can be simplified to read

$$F(k_x, k_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\, ds\, e^{-i2\pi kr} dr \qquad 2.9$$

The inner integral represents the original projection data (see equation 2.2) so equation 2.9 can be rewritten as

$$F(k_x, k_y) = \int_{-\infty}^{\infty} p(r,\phi)\, e^{-i2\pi kr} dr \qquad 2.10$$

If the transform of the projection data is represented by $P(k,\phi)$ then the one-dimensional Fourier transform of the projection data can be expressed as follows.

$$P(k,\phi) = \int_{-\infty}^{\infty} p(r,\phi)\, e^{-i2\pi kr} dr \qquad 2.11$$

Comparing equations 2.10 and 2.11, the right hand side of equation 2.10 represents the Fourier transform of the projection data which is defined in equation 2.11 as $P(k,\phi)$. This means that equation 2.10 can be further simplified to

$$F(k_x, k_y) = P(k,\phi) \qquad 2.12$$

Equation 2.12 is of fundamental importance and corresponds to the projection slice theorem. In words, equation 2.12 says that the one-dimensional transform of the projection at an angle ϕ corresponds to a slice taken at the same angle ϕ through the two-dimensional transform of the original distribution function $f(x,y)$. This is an important theorem because it means that a good estimate of $f(x,y)$ can be obtained from the projection data using

the following steps. First, a full set of projections is acquired. Second, the one-dimensional transform of each projection is obtained. Third, a two-dimensional array of Fourier coefficients is built from the one-dimensional transform of each projection by interpolation. Finally, an inverse two-dimensional transform is calculated to obtain the estimate of f(x,y).

The two-dimensional Fourier transform method of reconstruction works well but has a number of disadvantages. First, the reconstruction process cannot proceed until all the projections have been obtained. Second, the computation of a two-dimensional transform limits the speed of the reconstruction process. Hence, more computationally efficient methods of reconstruction such as backprojection are favoured.

2.3.2 Filtered backprojection

Simple backprojection represents a form of reconstruction that assigns a value to points in f(x,y) equal to the sum of ray sums passing through that point. This scheme is illustrated in figure 2.5. Mathematically, this guess at f(x,y), say $f_i(x,y)$ is given by

$$f_i(x,y) = \int_0^{\pi} p(x\cos\phi + y\sin\phi, \phi)\, d\phi \qquad 2.13$$

where a switch to the r,s co-ordinate system makes use of the relation given in equation 2.7. The integration in equation 2.13 accounts for the summation of each of the ray sums that contribute to a given point in the image. Although most of the qualitative information of the original attenuation map f(x,y) is preserved with simple backprojection, there are generally two

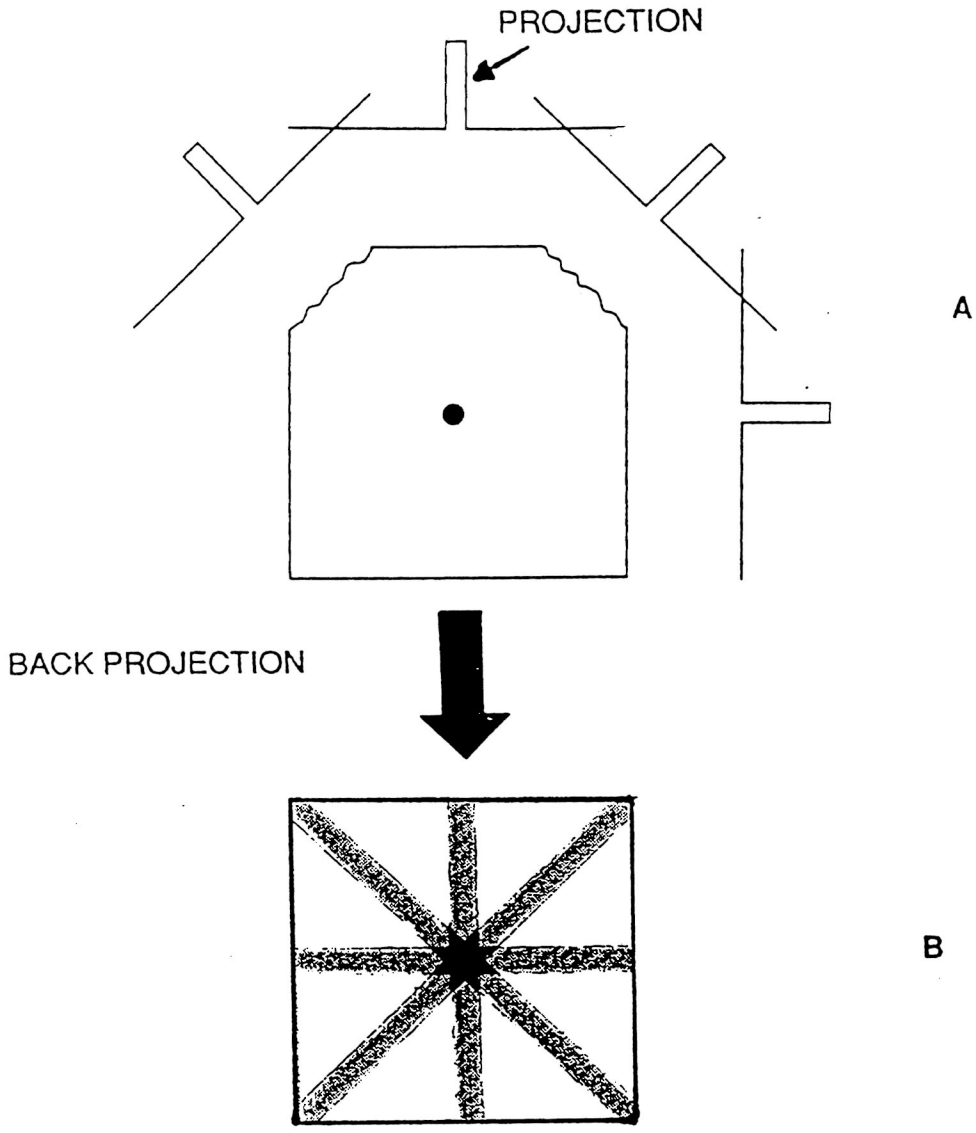


Figure 2.5: Simple backprojection. In (A), a series of projection profiles are shown graphically. In (B) these profiles are backprojected onto the image plane and summed to produce an approximation to the original object.

differences between the reconstructed image and the original. These differences are clearly evident in figure 2.5. First, many artifacts are generated in the reconstructed image. For example, star artifacts are common. Second, the relative intensities of the structures in the imaged cross-section are blurred.

Mathematically, it can be shown that the simple back projected image varies from the true image in that the reconstructed image is low pass filtered (Brooks and Di Chiro 1976). With the knowledge of the projection slice theorem expressed in equation 2.12, this low pass filtering effect can be quantitated in the following manner. If $F_i(k_x,k_y)$ represents the Fourier transform of the guess at f(x,y) obtained by simple backprojection, then the effect of low pass filtering can be expressed mathematically as

$$F_i(k_x,k_y)=\frac{F(k_x,k_y)}{|k|}=\frac{P(k,\phi)}{|k|} \quad 2.14$$

In words, equation 2.14 says that the image created by simple backprojection varies from the true image in that all frequency components are divided by the magnitude of the frequency. This suggests that backprojection can work if the projection profiles are properly filtered before being backprojected. Therefore, reconstruction by filtered backprojection requires the following three steps. First, a full set of one-dimensional projections is acquired. Second, each projection is filtered. Third, the filtered projections are backprojected to reproduce the original cross-sectional anatomy. There are a range of filtering functions which can be applied to the projetion data. These are described in the following section.

2.3.3 The filtering function

In order to reconstruct the original object with backprojection, theory (equation 2.14) requires that the projection profiles first be filtered by a function whose frequency spectrum is a linear ramp. This filter is sketched in figure 2.6. As shown, the ramp filter is designed such that the weight given to each frequency increases linearly as the frequency itself increases. If the projection profiles are simply backprojected as is, one would obtain a blurred representation of the original anatomy. The weighting scheme imposed by the application of the ramp filter compensates for this blurring by giving more weight to the higher frequency components of the image. The result is a sharper and truer representation of the original anatomy.

Before filtering, the already blurred projection data are also noisy due to the inherent Poisson statistical "noise". This noise spectrum has an approximately constant amplitude at all frequencies. The application of the ramp filter to the noisy projection data amplifies the noise component dominant at higher frequencies. The resulting reconstructed section is dominated by noise and appears very grainy. In theory, to prevent this, it is necessary to cut off the ramp filter near or at the point where the signal in the projection data disappears into the noise. An example of an appropriate choice of cut-off frequencies is shown in figure 2.7. In practice, the appropriate cut-off frequency is decided by the fact that the projection data is obtained at discrete intervals imposed by the physical size of the detectors, or collimators, and by the finite number of projections obtained

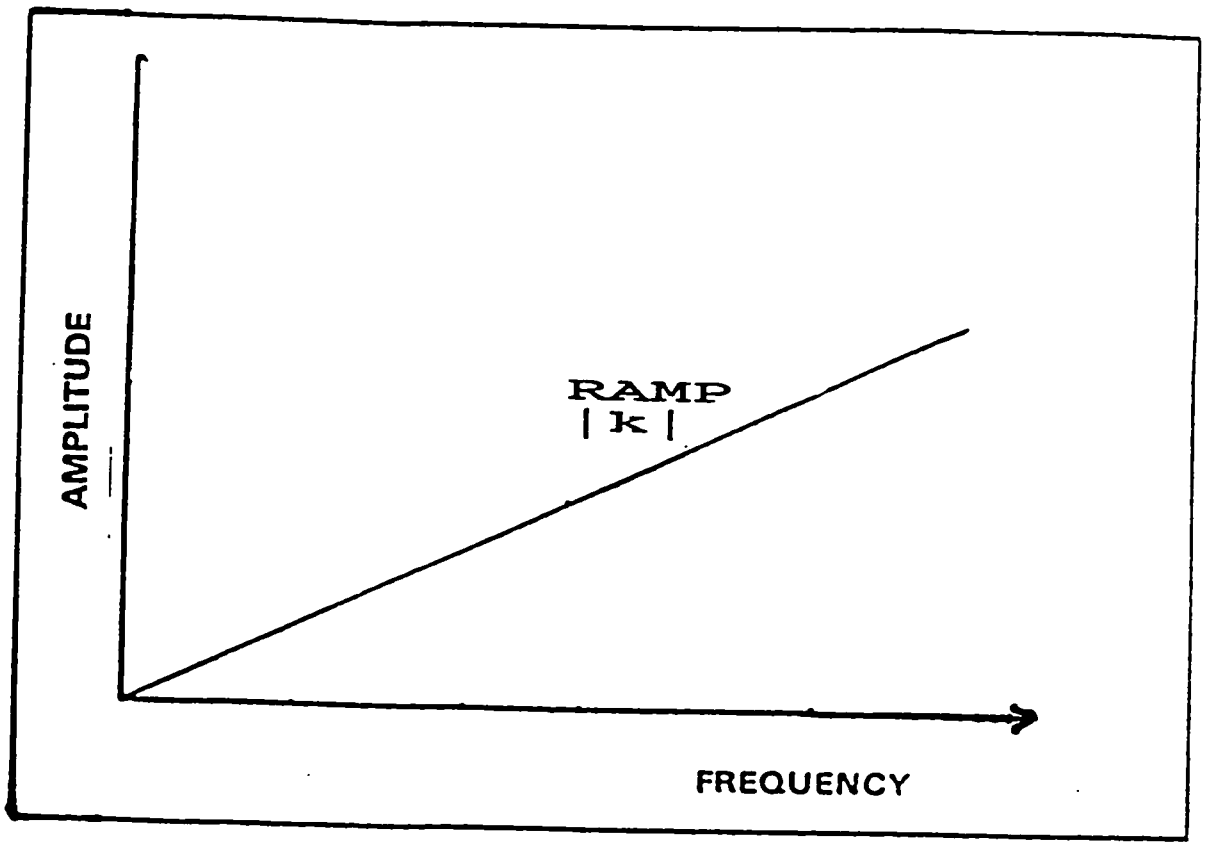


Figure 2.6: The ramp filter used to multiply the projection data before backprojection. The weight given to each frequency increases linearly with frequency.

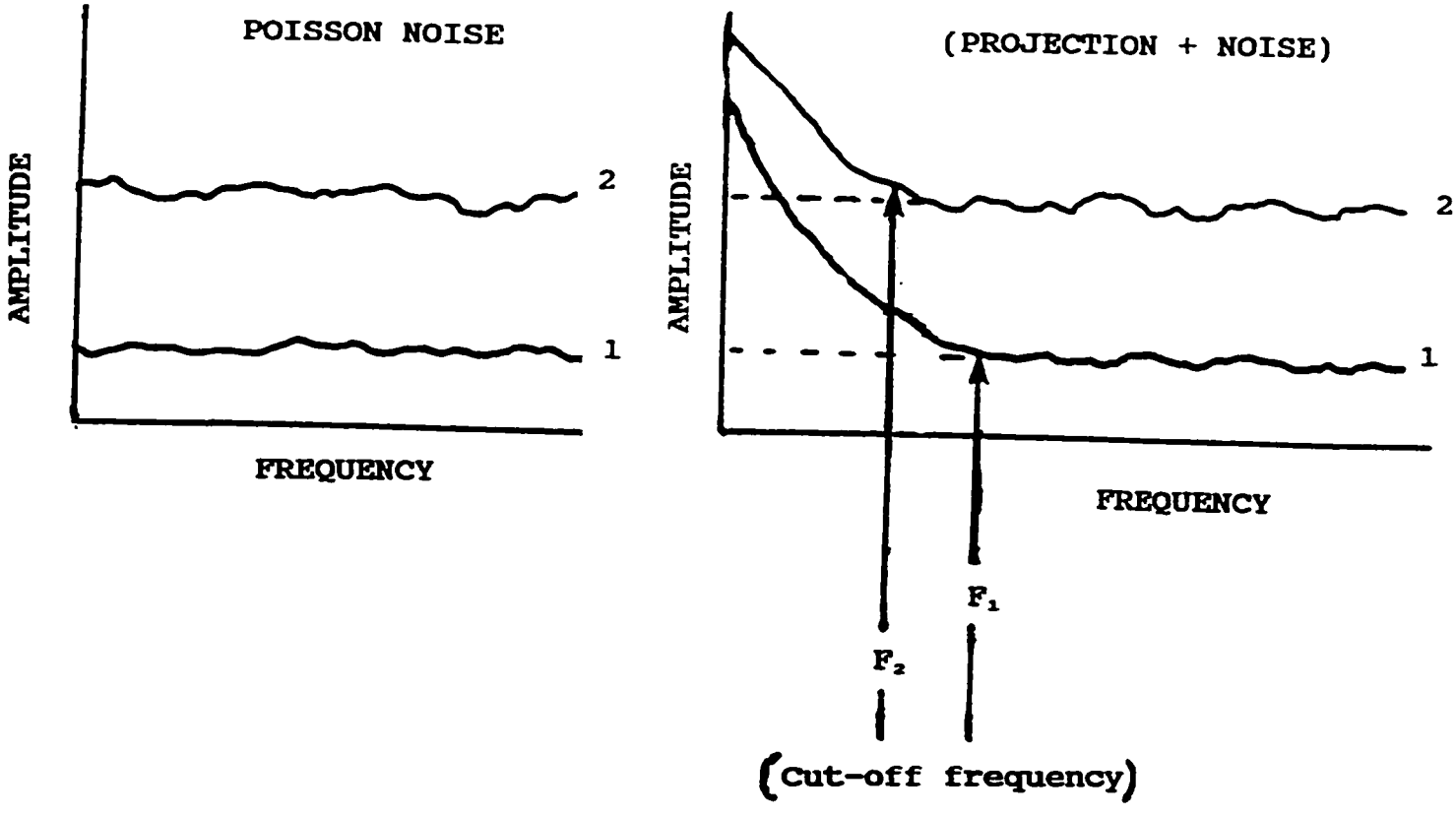


Figure 2.7: The appropriate choice of cut-off frequencies for the linear ramp filter. A range of frequency cut offs are indicated by (-----). The choice is defined by detector size and by the finite number of projections recorded around the object. If the detectors were infinitely small, then the choice of a frequency cut-off would be defined by the point at which the projection signal disappears into the noise. This is indicated by F_1 and F_2 which indicate the cut-off frequency corresponding to noise levels 1 and 2, respectively.

around the object. So, a higher cut-off frequency beyond that imposed by the finite detector size and the finite number of projections is wasteful because the resolution is still limited by the quality of the original projection data. The corollary of this last statement is that, unless the detector size is decreased and the number of projections increased, higher frequency cut-offs will only amplify noise without increasing resolution.

Frequency cut-off or windowing is achieved by multiplying the ramp function with a rectangular window represented by

$$\Pi(k) = \begin{array}{ll} 1 & -k_m < k < k_m \\ 0 & \text{elsewhere} \end{array} \qquad 2.15$$

and sketched in figure 2.8. The sharp frequency cut-off at k_m introduced by the rectangular window will give rise to oscillations at high frequency points such as sharp boundaries in the object. This effect is known as Gibbs phenomenon (Brooks and Di Chiro 1976) and can be reduced by using other types of frequency windows. Therefore, in practice, windows in which the frequency cut-off is rolled off rather than cut sharply are used. For example, the most commonly used window for filtered backprojection reconstruction is the window introduced by Hamming (1977).

$$\Pi(k) = \begin{array}{ll} \left|\alpha - [(1-\alpha)\cos(2\pi k)]\right| & -k_m < k < k_m \\ 0 & \text{elsewhere} \end{array} \qquad 2.16$$

$$\text{and} \quad 0 \le \alpha \le 1$$

α is the degree of freedom that allows for the matching of the filter to the noise and object detail. For example, setting α=1, the window becomes the rectangular window and it has the effect of maximizing resolution but with an appreciable increase in

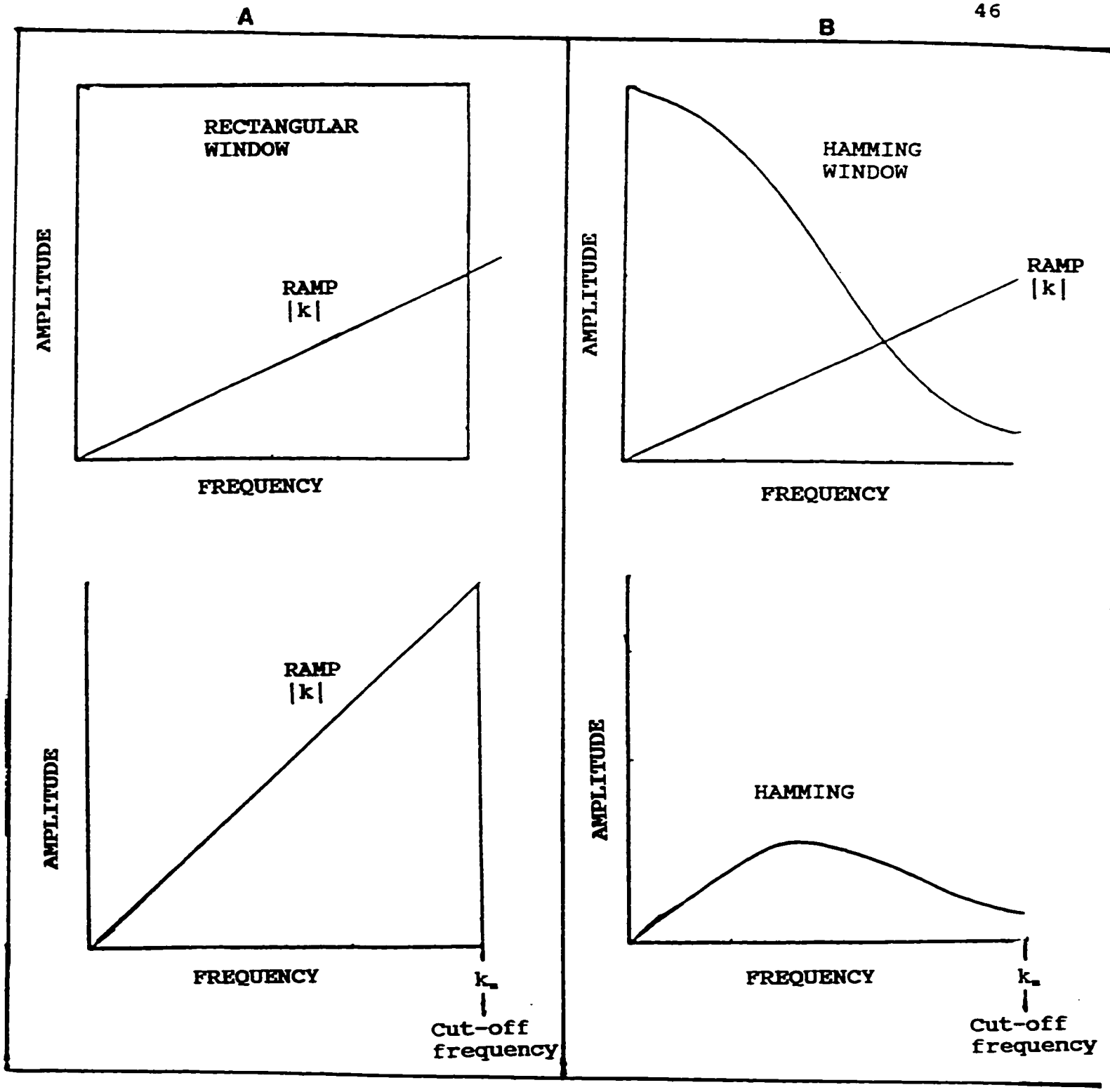


Figure 2.8: The windows indicated at the top of (A) and (B) are multiplied by the ramp function to produce the frequency domain filters sketched at the bottom of (A) and (B).

noise. Setting α=0, the window becomes a cosine window, a low pass window commonly utilized in signal processing to surpress noise but at the expense of an appreciable loss of resolution. The Hamming window with α=0.5 is also indicated in figure 2.8. It has the effect of being less sensitive to noise in the data, but will produce an appreciable loss of resolution.

The proper choice of the filter will depend on the noise level in the projection data and on the nature of the object being reconstructed. There is no universal filter. However, appropriate filtering can produce an image in which the recoverable object detail has been maximized, while the noise component has been minimized. The filter will be the ramp multiplied by a window appropriate to the object being investigated and tailored to match the noise characteristics in the projection data.

2.4 Limitations on resolution

The ability to resolve two adjacent features in a CT image is influenced by the degree of blurring and the noise level. The amount of blurring may be influenced by system geometric resolution limits, ray sampling frequency, pixel size, and properties of the convolution kernel applied before backprojection. Yester and Barnes (1977) have described the geometric limits of resolution as:

$$A_{eff}=\frac{1}{M}\sqrt{a^2+(M-1)^2 \cdot s^2} \qquad 2.17$$

where A_{eff} is the effective resolution in the image plane, a and s are the detector width and x-ray focal spot size, and M is the

geometric magnification at the centre of rotation. In most practical CT systems, the geometric magnification is constant. Then, it is apparent from equation 2.17 that resolution can be improved by reducing the detector width and focal spot size.

Spatial resolution is also influenced by the rate at which the projection data is sampled. The sampling theorem states that when one samples a continuous function such as a projection, spatial frequencies only up to a cut-off point are captured. This cut-off point is known as the Nyquist frequency and is defined as one half the spatial distance between samples. In other words, spatial frequencies whose period of repetition is less that two pixels cannot be visualized. To illustrate the sampling theorem, figure 2.9 shows the process of sampling a continuous projection profile at discrete points. Since the object is sampled at intervals equal to the distance between adjacent points in a projection, dx mm, the highest recoverable spatial frequency (k_m) is

$$k_m = \frac{1}{2dx} \quad mm^{-1} \qquad 2.18$$

The spatial resolution required to quantitate the structure of trabecular bone will be determined by the dimensions of the trabecular lattice. The column and strut model of trabecular bone assumes a column diameter of 0.2 mm and trabecular spaces of cross-sectional area of about 0.75 mm by 1 mm. Figure 2.10 gives this generalized treatment of the trabecular lattice and reveals that a spatial frequency of 1.05 cycles per mm is present in the trabecular lattice. Suppose one acquires images with a 10 cm field of view onto a 256 by 256 matrix. The sampling interval

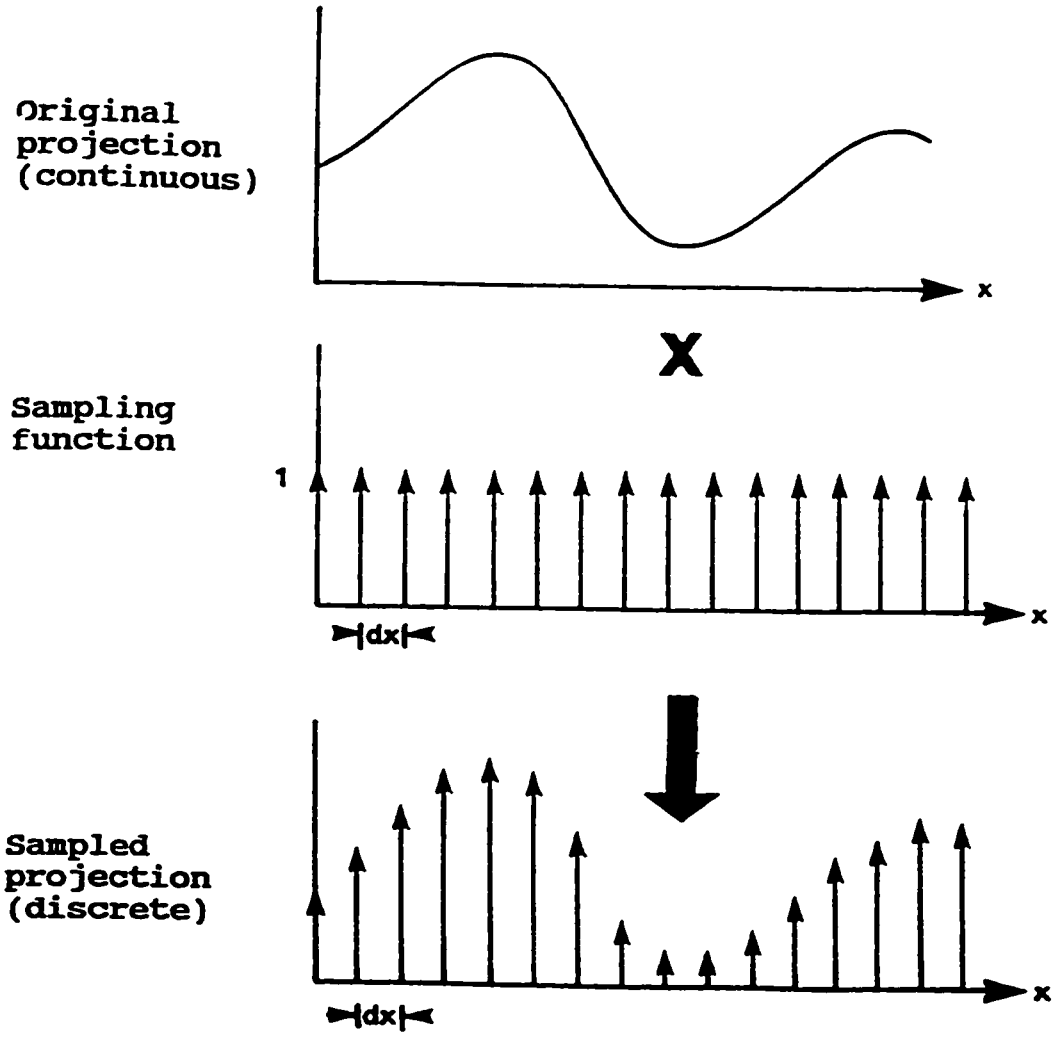


Figure 2.9: An analog projection profile is sampled. This sampling process is accomplished by multiplying the original projection with the sampling function. After multiplication, the shape of the sampled version is the same but is defined only at discrete points.

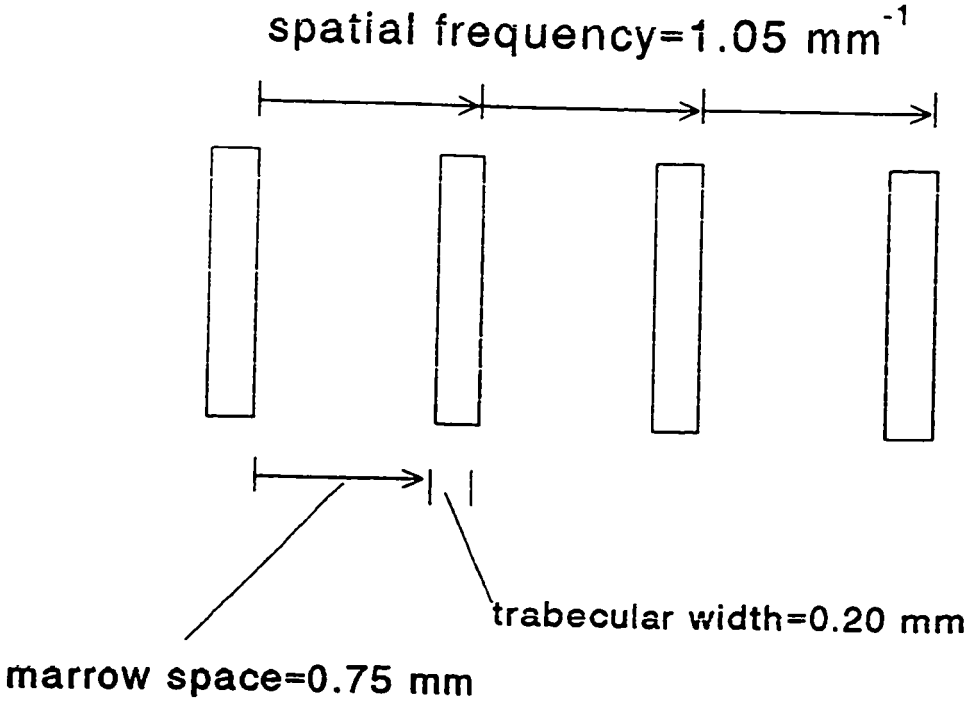


Figure 2.10: A representation of the spatial frequencies present in an average trabecular bone network.

across a horizontal line profile of pixels within the image would be 0.39 mm per pixel or simply 0.39 mm. Then, from equation 2.18, setting dx equal to 0.39 mm, the highest meaningful spatial frequency captured in the image matrix is 1.28 mm^{-1} . This is sufficient to reveal the idealized trabecular structure depicted in figure 2.10.

Noise levels also limit the ability to resolve two adjacent structures. The noise level in an image is influenced by pixel size (W) and the x-ray fluence (N). This relation can be expressed as:

$$Noise \propto \frac{1}{W \cdot N^2} \qquad 2.19$$

To improve spatial resolution the pixel size and the width of the x-ray beam can be reduced. This restriction in beam width and pixel size increases image noise. For example, if the pixel size is halved then the number of photons must be increased by a factor of four in order to maintain the same level of noise in the image. Also, if the width of the x-ray beam is halved, the x-ray fluence must be increased by a factor of two to maintain the same noise level.

Often the slice thickness in the z direction is several times greater than the pixel size in the x-y plane. This means that the resolution in the z direction is much worse than in the x-y plane. Partial volume effects will decrease the in-plane spatial resolution. In conventional CT, thick slices are necessary to keep statistical fluctuations small. For high-resolution applications the slice thickness should be reduced to minimize partial volume effects. The trade-off, however is

increased image noise. If the slice thickness is halved then twice as many photons are needed to prevent increasing the noise level. However, this requires that the scanning time be doubled and the subject receives twice the dose.

Chapter 3

PRINCIPLES OF MAGNETIC RESONANCE IMAGING

3.0 Introduction

Magnetic Resonance Imaging (MRI) is a method of obtaining cross-sectional images of the body by utilizing the magnetic properties of atomic nuclei. The manner by which an MR image is formed is different from that of standard x-ray imaging and Computed Tomography. What follows is an examination of the image formation aspects of MRI. First, the nuclear conditions which are necessary to produce an MR signal are examined. Second, the generation and measurement of the resonance signal will be discussed. Third, the means by which the emitted signal can be manipulated and arranged to form an image with varying degrees of contrast will be presented. Finally, the applicability of MRI to the study of in-vivo trabecular bone structure will be discussed.

3.1 Basic nuclear physics concepts

All atoms are made up of a central, massive, positively charged, nucleus around which orbit a number of negatively charged electrons. The nucleus itself is composed of nucleons: protons and neutrons. Since both protons and neutrons spin around their own axes, they possess spin angular momentum (S). Nucleons possess the same spin angular momentum as the electron (S=1/2 {spin up} or S=-1/2 {spin down}). One of the fundamental premises of electromagnetic theory is that any body that possesses both charge

and spin angular momentum will behave like a magnetic dipole. The strength and orientation of the magnetic dipole is quantitated by its magnetic moment. Which direction will be decided by the spin.

Nucleons occupy specific energy levels within the nucleus. In each energy level protons of opposite spin are paired and neutrons of opposite spin are also paired. However, protons with like spins, neutrons with like spins and protons-neutrons are not allowed to be paired. The up-down pairing results in a cancellation of the spin angular momentum. It is the spin angular momentum of the nucleus as a whole that determines magnetic resonance properties. If there is no net nuclear spin, there is no magnetic moment and hence no magnetic resonance properties. Since it is the number of unpaired nucleons that produce a net nuclear spin, the following rule defines which nuclei have magnetic resonance properties. Only those nuclei which contain an odd number of protons and (or) an odd number of neutrons will possess a net spin and be detectable by magnetic resonance. Some examples of elements that are measurable by magnetic resonance and are of clinical interest are $^{1}_{1}H_{0}$, $^{13}_{6}C_{7}$, $^{19}_{9}F_{10}$, and $^{31}_{15}P_{16}$.

It can be observed that protons have a magnetic moment by placing them in a static magnetic field. They attempt to align their magnetic moment along the lines of the static field- like a compass needle. Unlike a compass needle, because the proton is spinning, its magnetic moment will not align with the external field but will precess around it with an angular frequency (w_o). This alignment and precession of individual protons about the

static magnetic field is illustrated in figure 3.1. The frequency of precession is related to the static field strength (B_o) by the Larmor equation.

$$w_o=\gamma B_o \qquad 3.1$$

The proportionality constant γ is called the gyromagnetic ratio and is characteristic of the nuclei. For protons, γ is equal to 42.1 MHz/Tesla. Hence, if hydrogen nuclei(^1H) are exposed to a 1 Tesla static magnetic field, they will precess at a frequency of 42.1 MHz. If the field strength were doubled to 2 Tesla, then the precession frequency of the protons will also be doubled to 84.2 MHz.

Before the application of an external magnetic field, the magnetic moments of the protons in the body are randomly oriented. As shown in figure 3.2a their individual magnetic moment vectors point in all directions and their vector sum is equal to zero so that no net magnetic moment is produced. However, as shown in figure 3.2b, when a strong, static magnetic field is switched on, the orientations of the individual magnetic moments are no longer random. They will align with, or against the applied field. This alignment is discrete and can be viewed as two allowed states at slightly different energies. The ratio of the number of protons occupying each of the two states is defined by Boltzman's equation.

$$\frac{n(\uparrow)}{n(\downarrow)}=e^{-(\frac{\Delta E}{kT})} \qquad 3.2$$

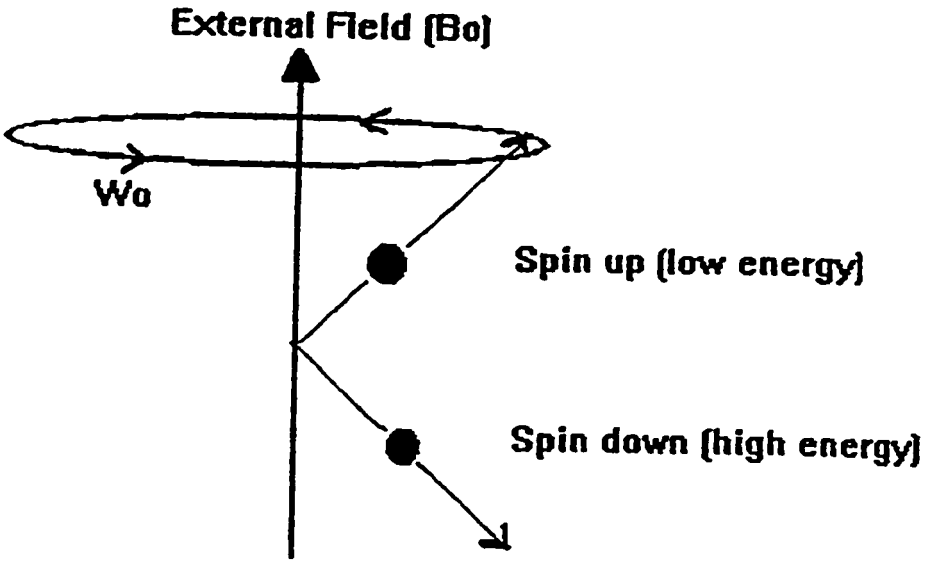


Figure 3.1: The spin up and spin down alignment assumed by nuclei in an external magnetic field. Each magnetic moment precesses at the Larmor frequency (W_o) about the external field. Although not drawn, nuclei in the spin down state precess in the opposite direction to those in the spin up state.

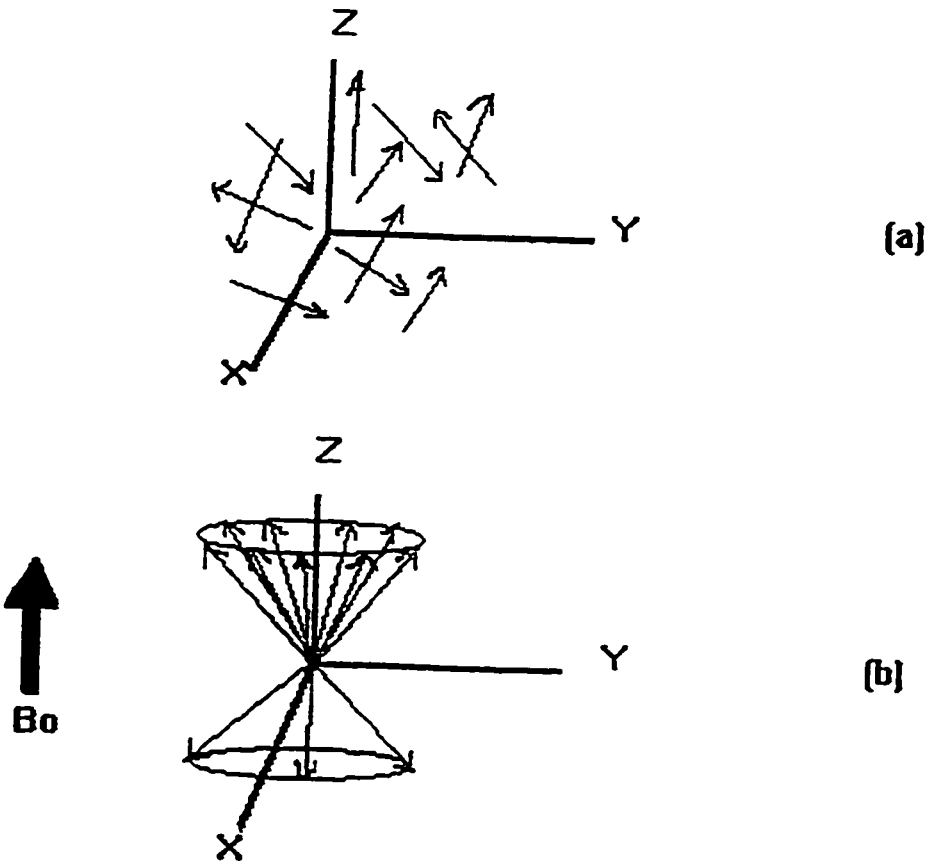


Figure 3.2: The ordering of the individual magnetic moments in response to the application of an external field B_o. As shown in (a), in the absence of an external field, the individual magnetic moments are randomly oriented and tend to cancel each other out. As shown in (b), following the application of an external field, the magnetic moments align themselves with or against the static field. Note that the magnetic moments precess about B_o and more are aligned with B_o than against it.

$n(\uparrow)$ and $n(\downarrow)$ represent the number of protons whose magnetic moment are aligned with and against the static magnetic field, respectively. ΔE represents the energy difference between the two energy states, T the tissue temperature, and k is Boltzman's constant. It is clear from equation 3.2 that more protons will align along the applied field that against it. This is illustrated in figure 3.2b which shows 9 nuclei in the low energy state and 3 nuclei at the higher energy configuration. Although only 12 nuclei are drawn here, in general for hydrogen based images of soft tissue, for every 10^6 nuclei in the down state there are (10^6+7) in the up energy state for the average body tissue (Schild 1990). Since each magnetic moment can be treated as a vector possessing both magnitude and direction, then the extra 7 moments per million will produce a net magnetization. More specifically, for the case sketched in figure 3.2b, the extra 6 moments in the up energy state will produce a net magnetization vector(M) along the Z direction, the direction of the static magnetic field. This result is drawn in figure 3.3a. The magnitude of this net magnetization vector is proportional to the number of protons in tissue. Because the phases of the precessing protons are random, the net magnetization vector does not have a component in the xy plane. This effect is illustrated by figure 3.3b. As shown, the "extra" magnetic moments precess randomly out of phase with respect to one another and so their components in the xy plane tend to cancel each other out. This leaves only the component of the net magnetization vector M in the Z direction.

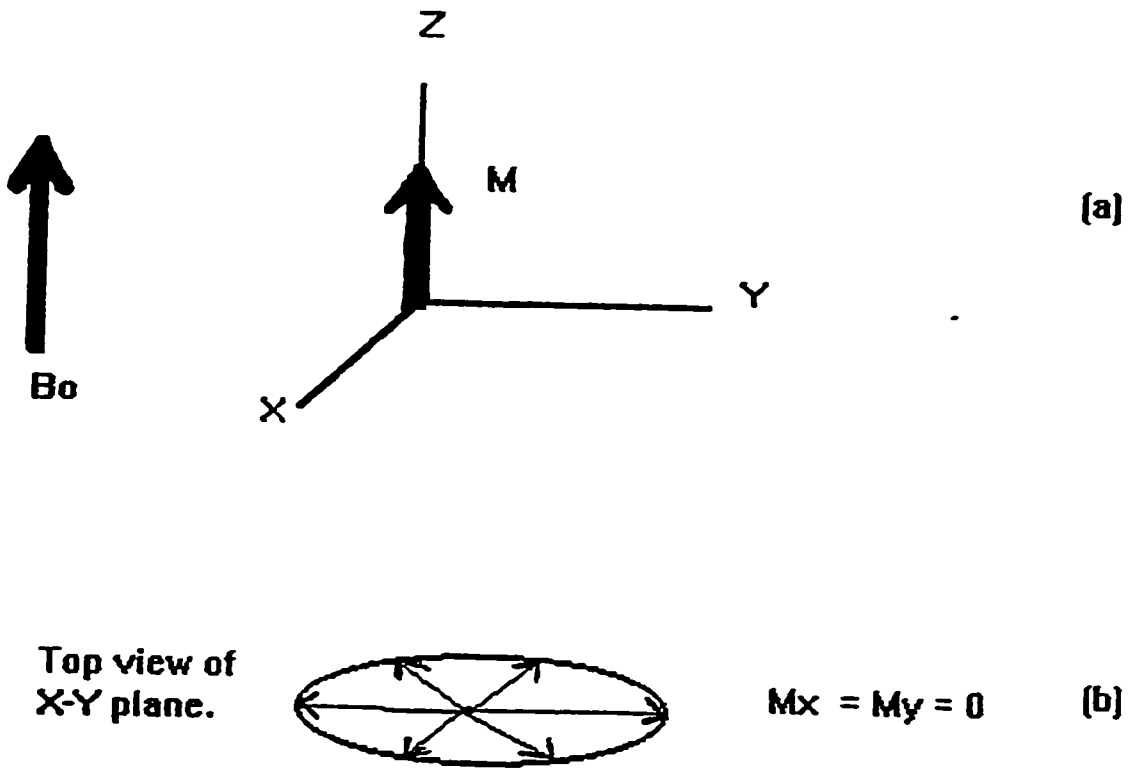
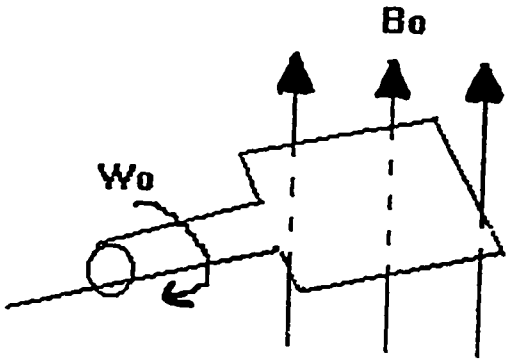


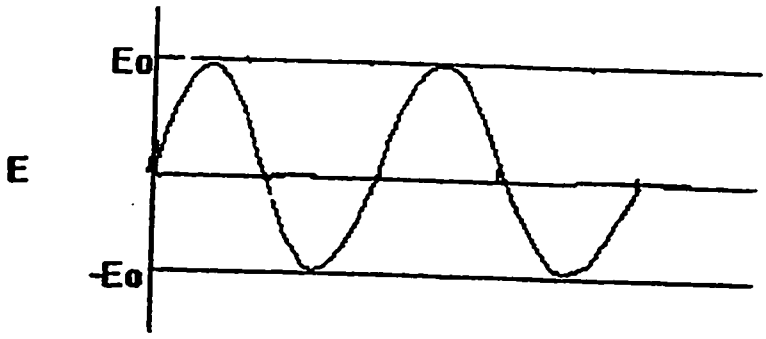
Figure 3.3: The establishment of a net magnetization vector M along z, the direction of the static field B_o (a). M only has a z component because, as shown in (b), the xy components of the individual magnetic moments tend to cancel each other out because the protons precess out of phase.

3.2 Sample excitation and signal generation

The measurable signal in magnetic resonance is an electrical current induced in a coil placed around the body part under study. This induction process is governed by Faraday's law as illustrated in figure 3.4 which shows a simple electrical generator consisting of a coil rotating with angular frequency w_o in a fixed magnetic field (B_o). As the coil rotates through the magnetic field the number of magnetic field lines crossing the coil's area at a given point in time changes. Faraday found that a changing magnetic flux induces an electromotive force (EMF) in the coil (Kane and Sternheim 1983). In the case of this simple generator, the induced voltage signal is sinusoidal with frequency w_o.

For signal acquisition in magnetic resonance the situation is reversed. The coil is fixed and the magnetic field is fluctuated. If a second magnetic field (B_1) is applied at right angles to the main field, two important effects occur. These two effects are depicted in figure 3.5. First, the magnetic moments of the individual protons are forced in phase, so that a component of M is created in the xy (transverse) plane. Second, the net magnetization vector will align itself along this new field, thereby being tipped into the transverse plane. The second magnetic field, B_1, is normally supplied by irradiating the tissues with RF radiation at the resonant frequency. Quantum mechanically this means that protons in the lower level (parallel to B_o) absorb energy, and are promoted to their excited state (antiparallel to B_o). The angle(θ) at which M is tipped from the z axis depends on the length of time





$$E = Eo\ Sin(Wo\ t)$$

Figure 3.4: An illustration of Faraday's law of induction for a simple electrical generator.

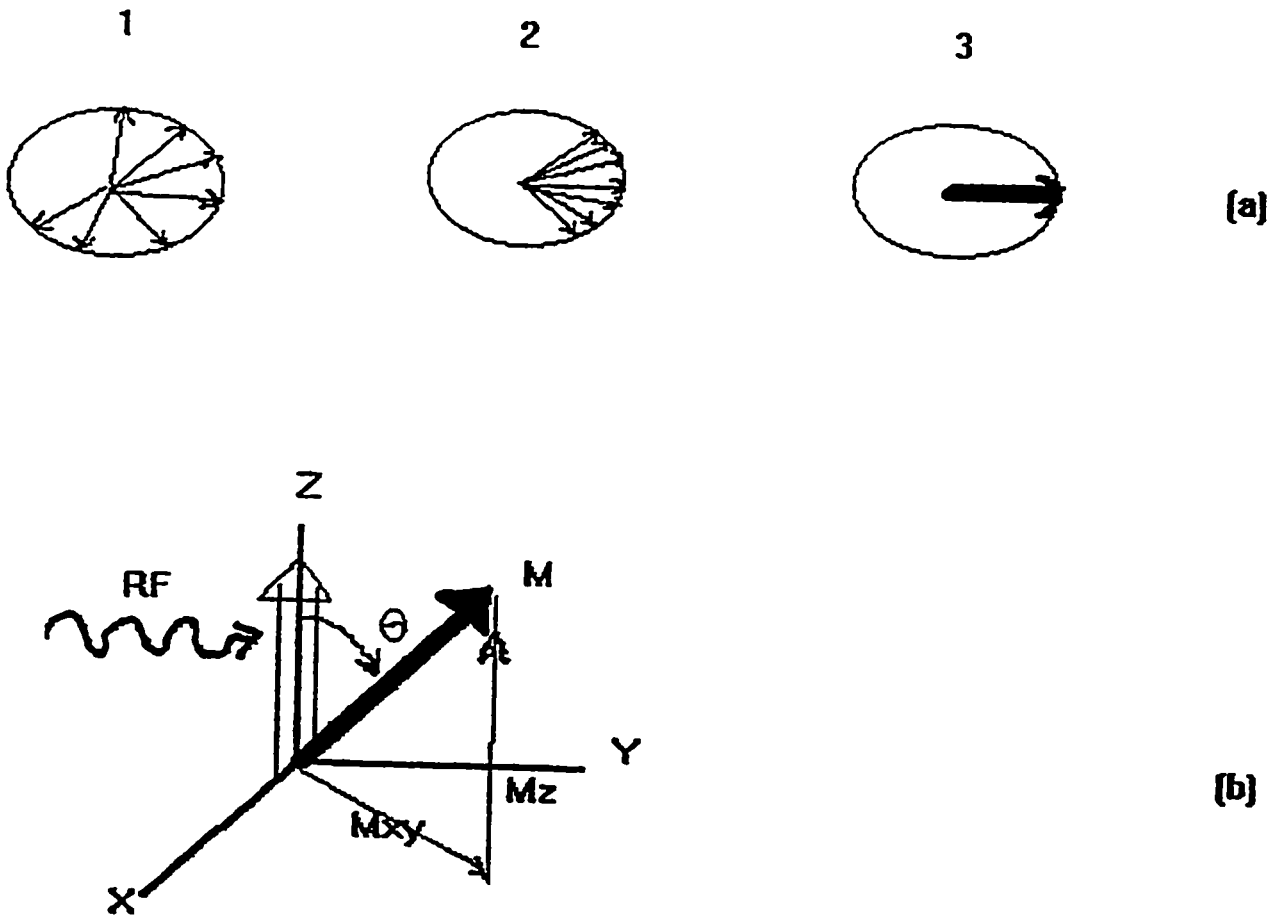


Figure 3.5: The perturbation of the net magnetization vector following an RF excitation pulse. First, the individual magnetic moments precess in phase to establish an xy component of M. Second, M tries to align itself along the new field B_1 generated by the RF pulse. In doing so M is tipped from the z axis through an angle θ.

(T_p) that the resonant RF pulse is on. This flip angle is calculated from equation 3.3.

$$\theta = k \cdot T_p \qquad 3.3$$

k is a constant related to the field strength. By changing T_p various flip angles can be achieved. Most commonly, M is flipped through 90° or 180°. In the case of a 90° pulse, the RF pulse is left on long enough that the difference in the number of nuclei at the two energy states equals zero and so the net magnetization vector will now exclusively precess in the xy plane. Following a 180° excitation pulse more nuclei reside in the high energy state than the low energy, hence M is flipped upside down.

When the RF pulse is switched off (B_1 is removed), the net magnetization vector will start re-aligning itself along B_o. In doing so, it will precess along B_o and represent a time varying magnetic field. An electric current will be induced in a nearby coil because of Faraday's law of induction. This induced current is the MR signal and is commonly referred to as a Free Induction Decay (FID). A typical FID is sketched in figure 3.6.

The rate at which the net magnetization changes with time can be defined by resolving M into its components along the x, y, and z directions. For example, the change in the net magnetization with time t after the 90° RF excitation pulse is switched off is given by the Bloch equation.

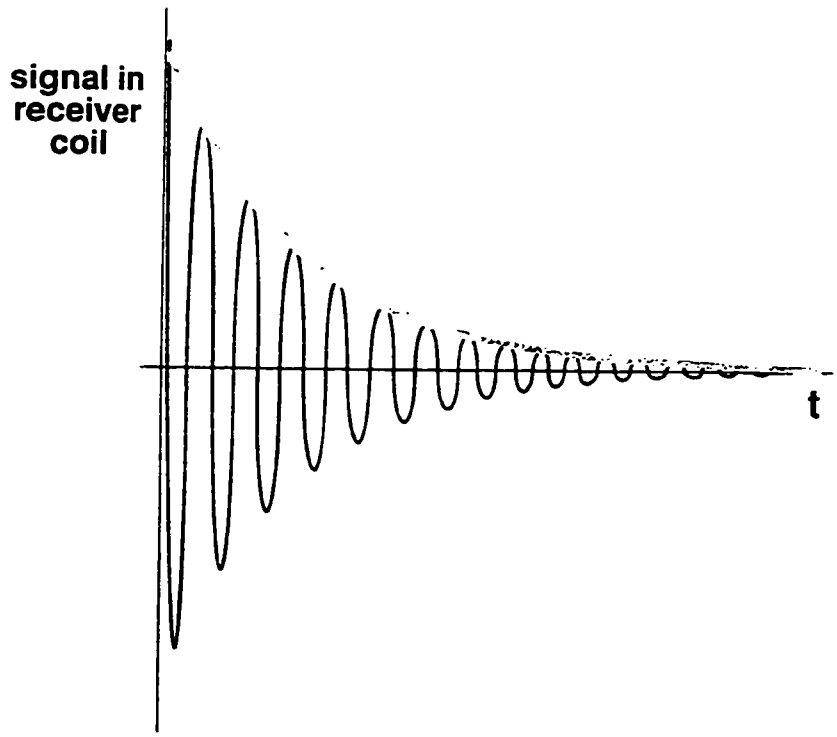


Figure 3.6: A typical Free Induction Decay signal recorded in the receiver coil following the application of a 90° RF pulse.

$$M(t) = M_o(1 - e^{-\frac{t}{T1}}) + M_o e^{-\frac{t}{T2}} \qquad 3.4$$

M_z and M_{xy} represent the vector components of the net magnetization vector along the static magnetic field (z) and transverse to it (xy), respectively. The decay constants T1 and T2 are the relaxation times and are unique for different tissues. The definition and role of T1 and T2 in MR imaging are explored in the next section.

3.3 Definition of T1, T2, and T2*

As defined by equation 3.4, there are two processes which contribute to M(t). The first is the return of excited nuclei to their low energy state. As more nuclei return to equilibrium with their environment, the difference in the number at the two energy levels approaches the equilibrium value predicted from Boltzman's equation (see equation 3.2). The result is the re-establishment of the z component of the net magnetization vector at an average rate of 1/T1. As nuclei return to their equilibrium state they give back their excess energy to the lattice surroundings as thermal energy. Therefore, T1 is commonly called spin-lattice relaxation time because it characterizes the time for the excited nuclei to realign themselves with the existing lattice structure of the material. T1 is also referred to as the longitudinal relaxation time because it is the time constant that describes the growth of the z component of the magnetization vector. A typical T1 relaxation curve is shown in figure 3.7. As indicated, T1 is not defined as the time required

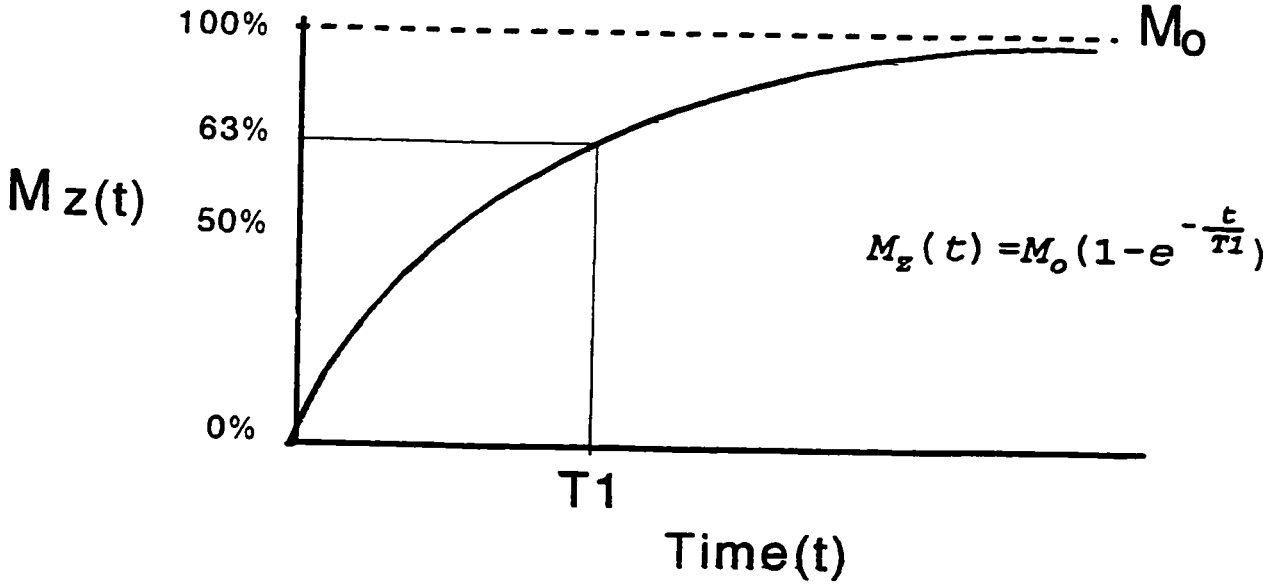


Figure 3.7: Growth of the z component of the net magnetization vector following a 90° pulse. This growth is exponential and is governed by the equation given. Note that T1 corresponds to the time required for the magnetization vector to recover 63% of its equilibrium value M_o.

for longitudinal relaxation to occur completely but is the time taken for the longitudinal magnetization to recover 63% of its original value. This result can be derived by setting t equal to T1 in the equation given.

The decay of the transverse component of the magnetization vector is due to inhomogenities in the magnetic field experienced by the nuclei. These inhomogenities arise from macroscopic and microscopic effects. The microscopic contribution to signal decay arises from field inhomogenities generated by molecular interactions and will dephase the nuclei with some protons rotating a bit faster than the resonant frequency, and others a bit slower. As the nuclei dephase, the xy component of the net magnetization tends to zero as the individual magnetic moments in the xy plane cancel each other. The rate governing this process is described by the T2 relaxation time. T2 is therefore referred to as the transverse relaxation time. A typical T2 decay profile is drawn in figure 3.8. The decay equation is also given. As indicated, T2 is defined as the time taken for the transverse component of the net magnetization vector to decay to 37% of its maximum (or decreases by 63%) value. This fraction can be verified by setting t equal to T2 in the equation given.

The macroscopic contribution to signal decay arises from inhomogenities in the applied magnetic field which will also dephase the nuclei. When combined with the microscopic contributions, this macroscopic contribution to signal decay is characterized by $T2^*$ to distinguish it from T2 which only accounts

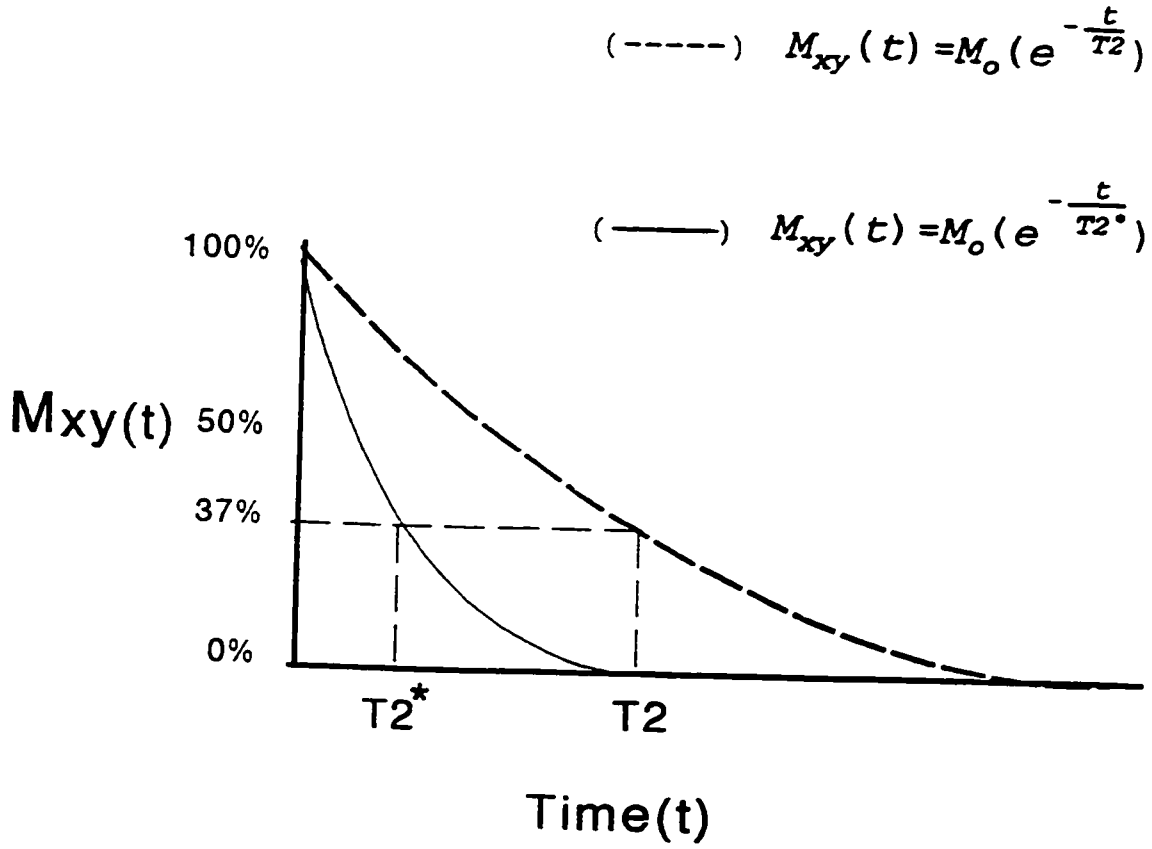


Figure 3.8: Decay of the transverse component of the net magnetization vector following a 90° pulse. T2 (-----) and T2* decay (-----) are shown. Decay is exponential and is governed by the equations given. Note that T2 corresponds to the time required for the magnetization vector to lose 63% of its initial value due to local field inhomogenities generated by the molecular environment. T2* corresponds to the time required for the magnetization vector to lose 63% of its initial value due to both local field inhomogenities generated by the molecular environment and by inhomogenities in the static field.

for the microscopic field inhomogenities. $T2^*$ decay is also sketched in figure 3.8. As shown, $T2^*$ will always be less than T2 because it accounts for all magnetic field inhomogentites. Because T2 effects are due solely to molecular interactions alone, T2 will depend more on the type of tissue being examined than T2*.

3.3.1 T1 and T2 for muscle, fat, and bone

For all biological tissue probed by magnetic resonance, T1 is greater than T2 and $T2^*$ is far shorter than T2. Typical relaxation times for muscle, fat, and bone are presented in table 3.1. The values listed are representative for the three tissues when imaged at 1.5 Tesla and can be used as a guide to understanding image contrast in MR. For example, image contrast can be optimized using T1 relaxation. Following a 90° excitation pulse, the weakest MR signal will be derived from tissues having the longest T1. This tissue will therefore appear dark in the image. It is quite clear from the T1 values given in table 3.1 that bone will always appear darkest in an MR image. Image contrast can also be optimized using T2 relaxation. The tissue with a longer T2 will produce a stronger signal and will therefore be brightest in the image. From the T2 values given in the table, fatty tissue will always appear brightest in an MR image and bone the darkest.

The extreme T1 and T2 values for bone are due to the strong influence that the molecular environment and temperature have on relaxation times. T1 and T2 are greatly dependent on the frequency of thermal motions among the molecules being imaged. This dependency

Table 3.1: Typical values for T1 and T2 for various body tissues.

Tissue Type	T1 (msec)	T2 (msec)
Water[a]	2700	2700
Muscle[a]	600	30
Fat[b]	250	50
Bone[c]	60000	0.1

Note: a- (Fullerton 1982)
b- (Dooms et al 1986)
c- (Ackerman et al 1992)

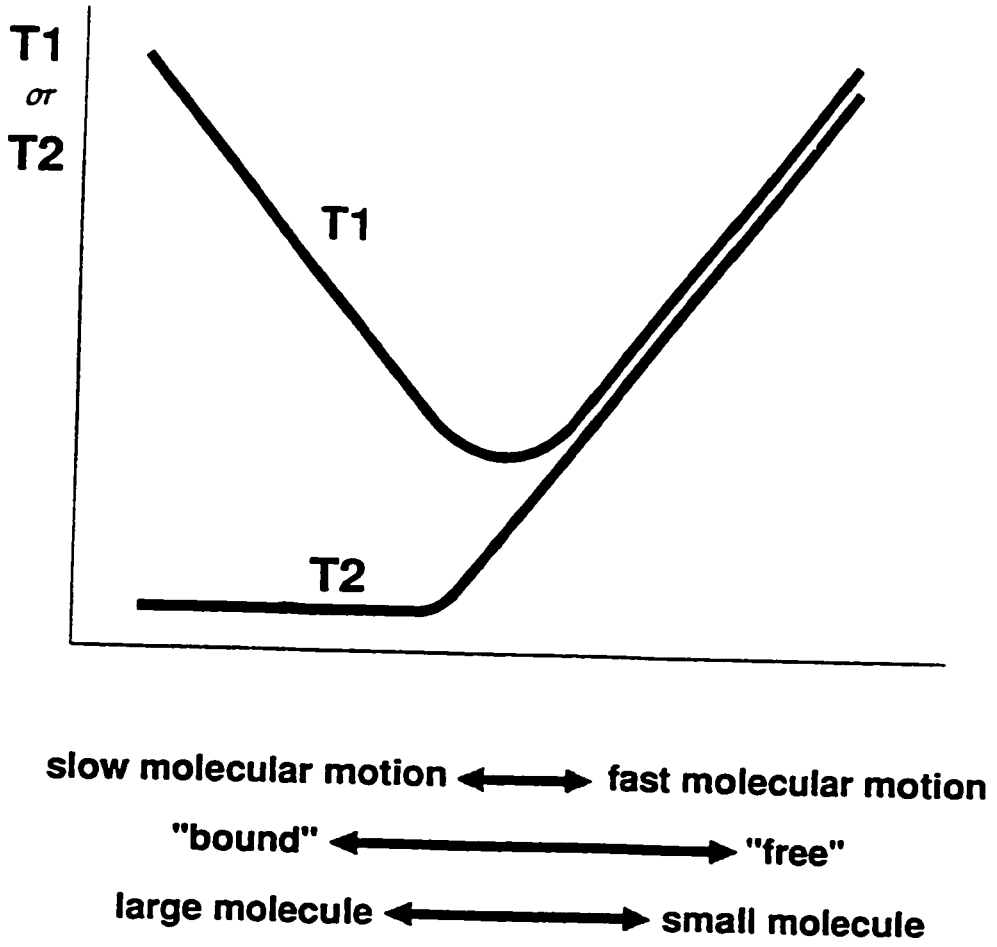


Figure 3.9: The dependence of T1 and T2 on the frequency of molecular motion and molecular size. The molecular environment corresponding to bone mineral will have a very short T2 and a long T1.

is sketched in figure 3.9. It is clear from these curves that a static and rigid structure such as bone does not provide the "mobile" protons necessary for MR imaging. Alternatively, tissues with a high water content such as muscle, will produce a strong MR signal.

3.4 The spin echo and gradient echo pulse sequence

Following the application of a 90° pulse the FID signal quickly disappears as the protons dephase due to magnetic field inhomogenities (T2* effects). The application of a 180° pulse has the effect of reversing the magnetic moments of the protons which reverses the dephasing, thereby "refocusing" the protons and producing an echo (a measurable signal). To illustrate this dephasing and subsequent echo, consider the following numerical example. Suppose that there is a nucleus precessing with a frequency of 10 MHz. Due to local inhomogeneities, one of its neighbouring nuclei is in a field which is 1% stronger. This means that the neighbouring nucleus precesses at 10.1 MHz. In 5 microseconds, the nucleus precessing at 10 MHz turns 50 times while the other rotating at 10.1 MHz has turned 50.5 times. So in 5 microseconds, the two nuclei are exactly 180° out of phase, cancelling their magnetic moments in the xy plane. Now, if a 180° pulse is applied, the spins of the protons will be reversed. This means that the proton precessing at 10.1 MHz will return "in phase" with the proton precessing at 10 MHz approximately 5 microseconds after the application of the 180° pulse. Generalizing from this

example, the application of a 180° pulse a time τ after the 90° pulse produces an echo following a time 2τ after the 90° pulse. This is the spin echo pulse sequence and is sketched in figure 3.10. The timely application of the 90° and 180° pulses can be repeated with a time TR. The echo time (TE) indicates the time between the 90° pulse and the peak of the signal echo. TR is always much longer than TE, although it does not appear to be so in this figure.

The production of an MR image using the conventional spin echo pulse sequence can be a fairly slow process. To decrease imaging time fast imaging techniques which use short repetition times and excitation RF pulses less than 90° can be used. One such fast scanning technique is the gradient echo pulse sequence. A basic gradient echo pulse sequence is sketched in figure 3.11. This sequence differs from the spin echo sequence of figure 3.10 in that the 180° echo pulse is missing, the 90° pulse has been replaced by an α pulse, and two x-gradient pulses, each opposite in sign, have been added. α indicates the angle through which the magnetization vector is tipped away from the z axis and is the same angle calculated by equation 3.3. After an excitation pulse of α (less than 90°), a positive longitudinal magnetization (M_z) is still present. If a 180° pulse is then applied it would invert M_z, thereby moving it far from equilibrium and nullifying the advantage of the short TR achieved by using a small flip angle. Therefore, it is inappropriate to combine the very short TR and small flip angle with a 180° pulse sequence to produce an echo. Instead, an echo is obtained by means of a gradient reversal. Immediately after the

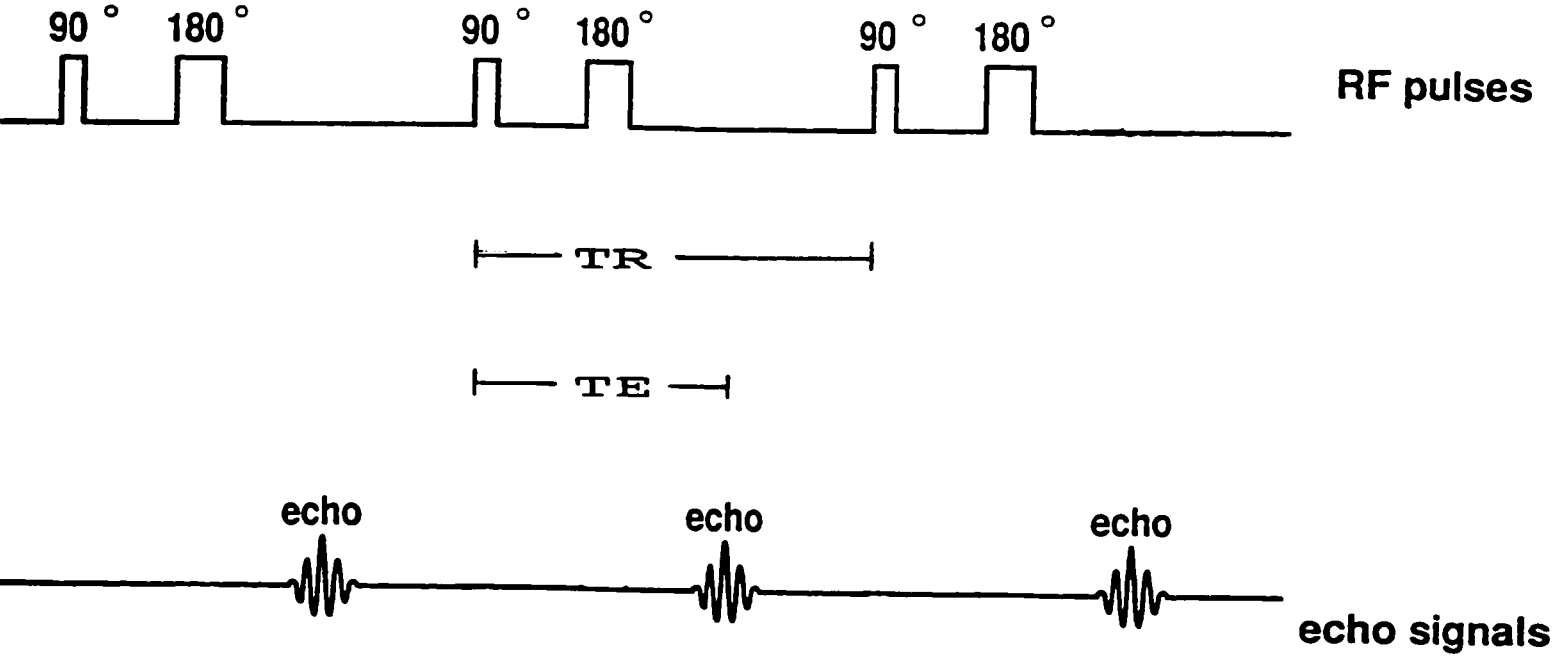


Figure 3.10: A basic spin echo pulse sequence. In this diagram, 3 cycles of the pulse sequence are shown. In general the pulse sequence is repeated several times for improved signal to noise. The timing of the sequence is defined by TR and TE.

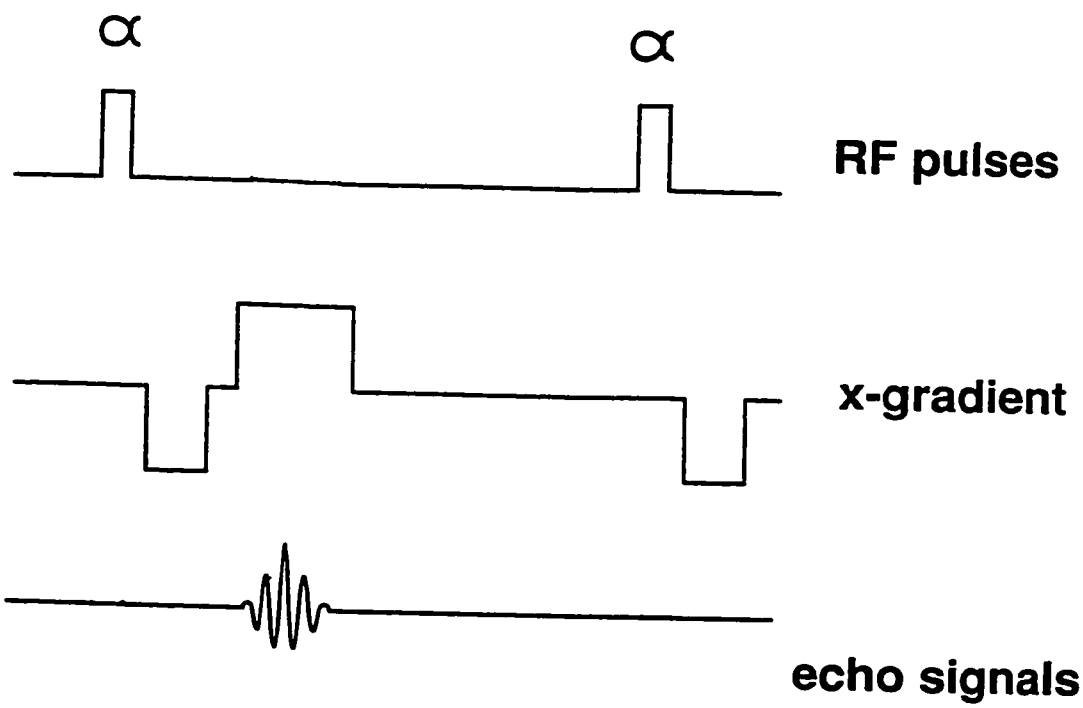


Figure 3.11: A basic gradient echo pulse sequence. An RF pulse of angle α, less than 90°, is used to excite the protons. The echo is generated solely by the x-gradient.

excitation pulse, a negative x-gradient is applied. During this gradient, the protons precess with a different frequency that depends on their location along the x-coordinate. Hence, the effect of this negative gradient is a dephasing of the protons. The x-gradient is then reversed, thus switching the direction of increasing magnetic field strength along the x-coordinate. The protons that had the fastest precession during the negative gradient now have the slowest precession, and vice versa. The effect of this positive gradient is a rephasing of the protons. This rephasing results in a gradient echo. The absence of the 180° pulse means that dephasing due to magnetic field inhomogenities is not cancelled out at the time of the gradient echo. This reduces the size of the echo because the signal is strongly influenced by $T2^*$ effects rather than T2 effects.

3.5 Image Formation

Image formation requires the spatial localization of the MR signal. This localization is a three step process which is shown in figure 3.12. First, a slice selection gradient is switched on during the application of the RF excitation pulse. This gradient, G_z, is applied along the direction of the external magnetic field B_o. Therefore, only those protons within a given slice Δz will be excited by the RF pulse with frequency

$$w=\gamma (B_o+G_z) \qquad 3.5$$

The location and thickness of a slice can be varied by selecting

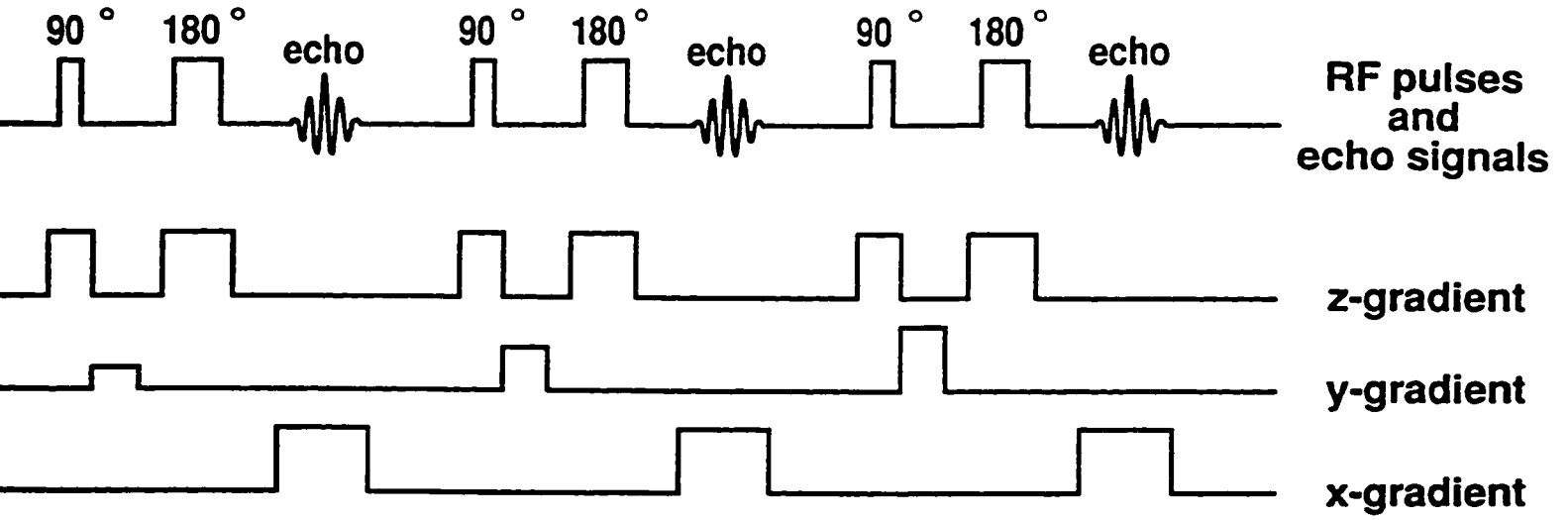


Figure 3.12: Image formation using a basic spin echo pulse sequence. In this diagram, 3 cycles of the pulse sequence are shown. They are all identical except that the phase encoding y-gradient is increased step-wise for each pulse sequence. The 90° RF pulse, the 180° RF pulse, and the MR signal appear together on the top line.

the frequency and bandwidth of the RF pulse. To generate spatial information within a selected slice, magnetic field gradients are applied in the x and y direction. When the RF pulse and slice selection gradient G_z are switched off, a y gradient is turned on for a brief time. This G_y produces a phase shift in the angle of the precessing protons and magnetization in each voxel. This phase shift varies with the y position of the voxel. Consequently, immediately after G_y is turned off there will be a linear relationship between the phase angle, produced by G_y, and the y-coordinate of the voxel. The y direction is often referred to as the phase encoding direction. After G_y is switched off, the 180° rephasing pulse is applied. After the 180° pulse, the x gradient is turned on for the entire duration of the echo signal. Thus, during the echo signal, the precessional frequency of the magnetization in each voxel is determined by the x-coordinate of the voxel. Since G_x is on while the echo is being detected or "read", it is also called the readout gradient and the x direction is referred to as the frequency encoding or readout direction.

To generate a single MR image with a 256 x 128 matrix, 128 echo signals must be obtained, each from a separate pulse sequence having a different y-gradient strength. Each of the 128 different signals produced must be sampled at 256 equally spaced times (along the frequency direction). In figure 3.12, 3 cycles of the pulse sequence are shown. They are identical except for the change in the strength of the phase encoding gradient (G_y). For a 256 x 128 image matrix G_y is increased step-wise for each of the 128 pulse

sequences. As each echo is "read out" it is sampled at 256 points in time. If a 1-dimensional Fourier transform is performed on the 256 data points from the echo, the result will be 256 amplitudes and phases, each representing a sine wave of a different frequency. Each of these sine waves is the sum of the MR signals from all voxels having the same x-coordinate because a point along the x-coordinate can be identified by a unique frequency. If all of the 128 signal echoes are considered then the result is a matrix containing 32,768 data points. Again, if a 2-dimensional Fourier transform is taken of this data set, the result is the image matrix of 256 x 128 pixels. The signal intensity of each pixel in this image matrix is proportional to the MR signal that originated from within the voxel of tissue associated with the image pixel.

3.5.1 Scan timing

A given pulse sequence needs to be repeated many times in MR imaging to produce an image matrix of a selected size. The number of repetitions depends on the number of pixels in the phase encoding direction (y-direction). If 256 pixels are wanted in the y-direction, 256 repetitions of the pulse sequence and step-wise increases of the y gradient must be performed. If 128 pixels are wanted along y then 128 repetitions of the pulse sequence are needed. Since it takes a given time to complete the cycle of a given pulse sequence, it is clear that it will take twice as long to produce an image with 256 points along the y direction than 128.

To increase signal to noise, each pulse sequence can be

repeated one or more times before changing the y-gradient. The number of pulse sequences applied with the same y-gradient strength is referred to as the number of excitations (Nex). If an image with 128 points along the y direction is acquired at 2 Nex then the pulse sequence will need to be repeated 256 times. If 4 Nex is needed to obtain the desired signal to noise ratio the y-gradient is increased one step with every fourth pulse sequence. In this case the total imaging time will be increased by a factor of four with respect to an image acquired at 1 Nex.

It is obvious that due to the large number of repetitions, the acquisition of an MR image can be a lengthy procedure. The total acquisition time is equal to the product of the repetition time (TR), the number of excitations (Nex), and the number of phase encoding steps (number of pixels in the y-direction). The mathematical computation of the 2-dimensional Fourier transform needed to reconstruct the image takes only a few seconds.

3.6 Evaluation of trabecular bone quality

The main challenge in the quest to assess trabecular bone structure in-vivo is the achievement of MR images with sufficient signal to noise (SNR). Signal to noise in MR is affected by three critical parameters. They are sample size, spatial resolution, and magnetic field strength. Because SNR is inversely related to sample size, obtaining images of sufficient resolution is only possible for selected anatomical sites such as the wrist, phalanges, and calcaneus. These peripheral sites permit the use of small, tightly

coupled RF surface coils which can be placed close to the anatomy of interest. For example, the SNR achievable in the wrist (imaged at 6 cm field of view) is about a factor of 15 worse than that achievable at the phalanges (imaged at 2 cm field of view) (Wehrli et al 1993). It is evident that for larger fields of view such as the spine and the hip, poor SNR will preclude measurements of structure. To obtain images of sufficient resolution and SNR one has to aim for small anatomical sites and to balance the requirements for increased SNR and spatial resolution.

One of the most critical requirements to be met for in-vivo images of trabecular structure is an imaging slice thickness comparable to the mean thickness of structures to be differentiated. For trabecular bone the mean thickness of individual trabeculae is approximately 200 μm. The slice thickness achieved on current whole body clinical systems is almost a factor of three greater. For relatively isotropic trabeculae the imaging slice thickness is critical. However, thicker slices may be allowable if image processing steps can extract structural information from these volume averaged images.

MR provides a high degree of contrast between mineralized bone, which has background intensity, and the protons in the fat occupying the intertrabecular space from which the MR signal is derived. As such, an MR image of trabecular bone structure is uniquely suited for digital image processing. MR also permits the examination of structure in any of the three orthogonal scan planes. Unlike CT, MR provides these images without the use of

ionizing radiations. Coupled with the ability to interrogate structure in each of the three scan planes, the absence of ionizing radiation makes MR an attractive tool for assessing trabecular bone structure in-vivo.

Chapter 4

DESCRIPTION AND CHARACTERIZATION OF THE XCT 960

4.0 Introduction

The STRATEC pQCT scanner series originates from the University of Wurzburg in Germany where it was developed in close cooperation with Stratec Medizintechnik (Germany). The XCT-960 is one such scanner which is marketed in North America by Norland Corporation (Fort Atkinson, Wisconsin USA 53538). In its standard clinical version, the XCT 960 obtains a CT image with a 128^2 matrix and a minimum pixel size of 0.59 mm. There is little structural information which can be extracted from this standard image and so a non standard version of the scanning software has been developed by STRATEC. This updated software enables the acquisition of the CT image with improved resolution onto a larger image matrix size. The improvement was achieved by doubling the number of projections obtained at the expense of increasing the scanning time. This non-standard version of the software has been distributed to a limited number of centres for use.

In this chapter the technical specifications of the pQCT scanner are examined. The improved scanning software is described. Finally, the suitability of the images for assessing trabecular bone structure is considered on the basis of image noise, image contrast and spatial frequency limits.

4.1. Description of Scanner

The XCT 960 is a second generation scanner which implements the fan beam translate-rotate principle to acquire transaxial images of the distal end of the radius. A schematic of the scanner is sketched in figure 4.1. As shown, it employs 5 CdTe detectors to acquire the necessary projections over 180°. The sixth detector is used to acquire the scout view scan. The slot size in the detector collimator screen is (0.3x3.5) mm. Collimation of the x-ray tube output produces a 2.5 mm slice thickness at the centre of rotation. The tube is operated at 45 kVp but 5 mm of aluminum filtration produces an x-ray beam with a mean energy of 40 keV and a full width half maximum of 8 keV. This energy is optimal for evaluating small changes (1%) in trabecular bone density of the peripheral skeleton for a low patient skin dose (0.2 mSv) (Muller et al 1985). During rotation the projection data set is obtained through 29 angular steps (each 6.25°) of the detector array around the object. During translation each of the five detectors records a projection. As a result, 145 projections are obtained at 1.25° angular steps around the object. Each projection is sampled at 0.33 mm intervals and 254 samples are taken per projection. Images are reconstructed using filtered backprojection employing the convolution kernel described by Shepp and Logan (Shepp and Logan 1974). The image is bandlimited during reconstruction. This allows for the image to be reconstructed onto an array with a pixel size equal to the sampling interval of 0.33 mm and makes the assumption that the imaged structure does not contain spatial frequencies

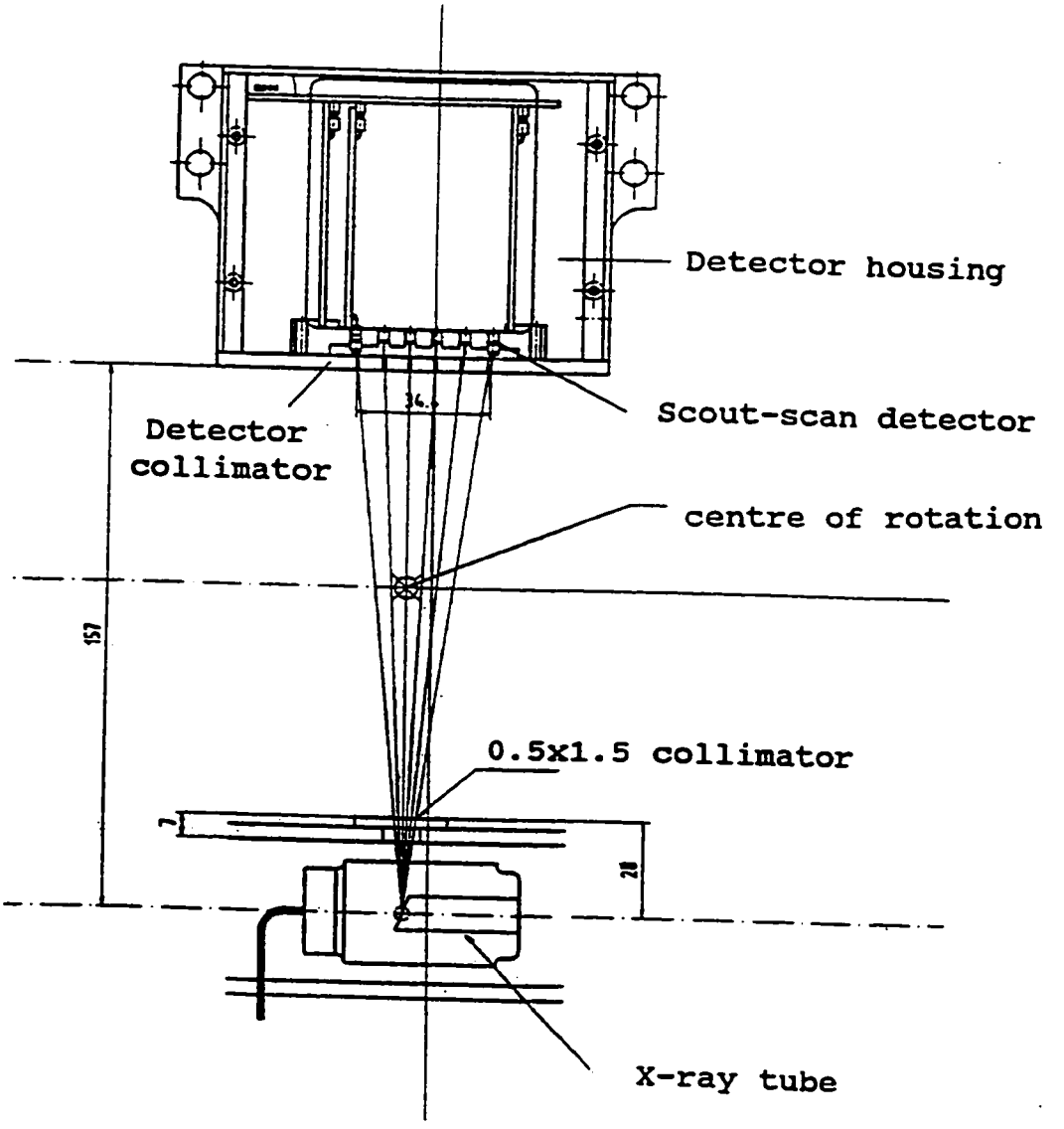


Figure 4.1 A cross-sectional representation of the geometry of the XCT 960 pQCT scanner. All distances are labelled in millimetres.

greater than 0.5 mm^{-1}. As sketched in figure 4.2, this is a valid assumption. The final image is displayed as a matrix of linear attenuation coefficients which are scaled by 1000. The voxel values are linear attenuation coefficients and are not defined as Hounsfield numbers which relate the measured linear attenuation coefficient to that of water. Therefore, throughout discussions of the pQCT image, the voxel values will be referred to as linear attenuation coefficients (LAC).

4.2. Image Acquisition and Quantitation.

The gantry of the XCT-960 can move in discrete 1 mm steps along the object to be scanned. With this in mind, a typical pQCT examination follows three basic steps. First, a coronal computed radiograph (scout view) of a 30 mm section encompassing the distal end of the radius is obtained as the gantry moves through 30 discrete steps. The appearance of the scout view is sketched in figure 4.3. Second, as indicated, the head of the radius is marked with a reference line and the scanner gantry moves a fixed distance proximal and along the subject's arm from the marked position. Third, the high-resolution CT scan is recorded at this proximal site. To ensure direct comparison between subjects, the distance moved from the head of the radius by the gantry is fixed at 4% of the subject's arm length (ulna length). The total scanning time required to acquire both the scout view and cross-sectional image is 10 minutes. A typical pQCT image obtained after reconstruction is shown in figure 4.4. This is a scan of the left arm of a normal

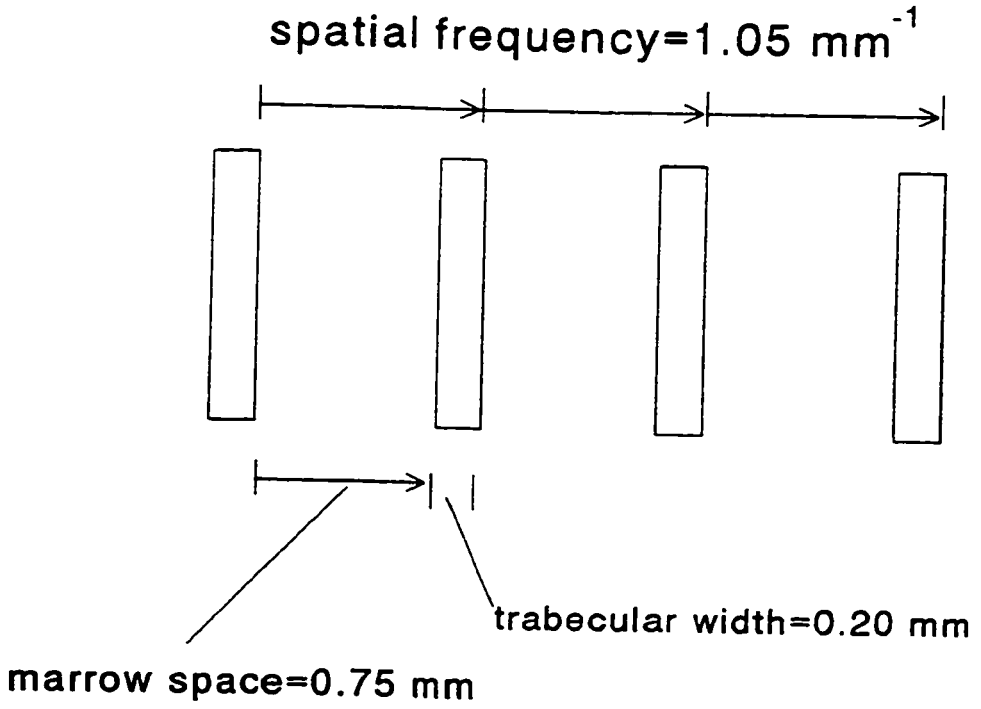


Figure 4.2 Schematic of the spatial frequencies present in the average trabecular bone network with an average trabecular width of 0.2 mm and an average marrow space of 0.75 mm.

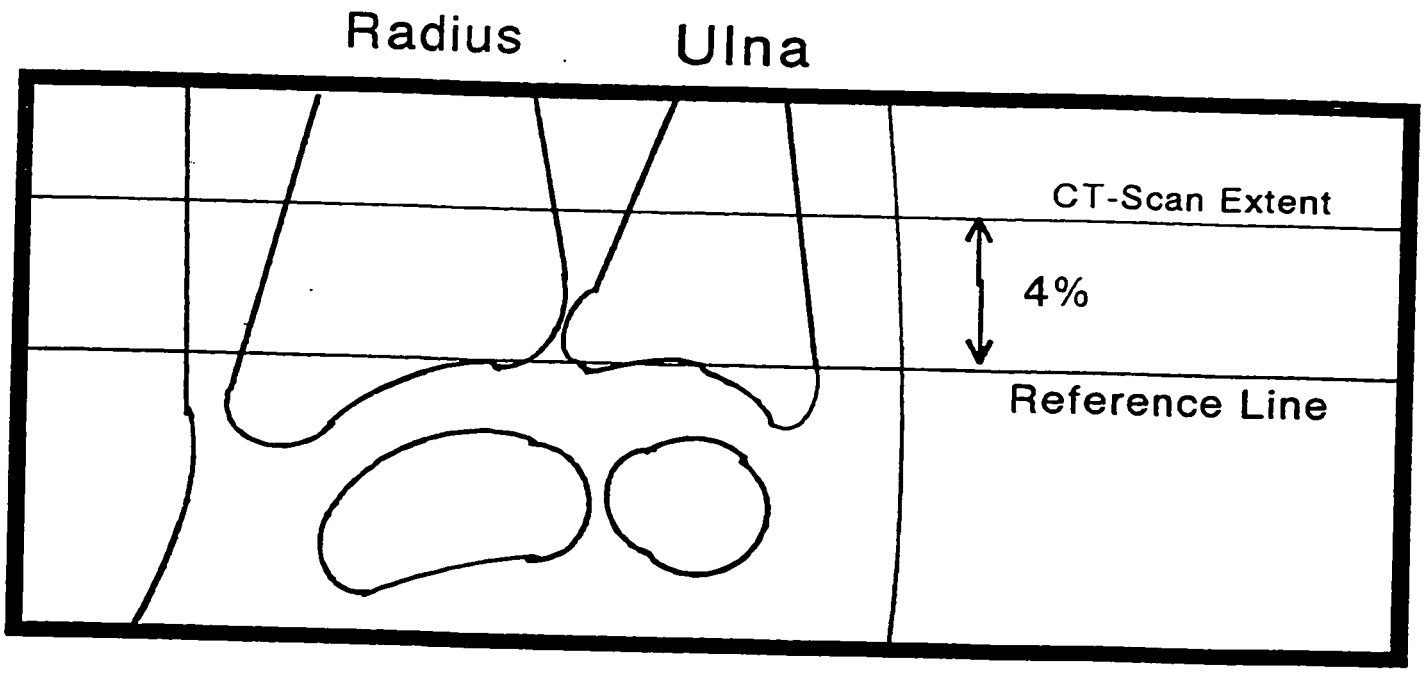


Figure 4.3 Scout view representation of the anatomy at the distal end of the radius.

23 year old female volunteer. The bone which appears largest on cross-section is the radius and is the focus of analysis because it represents a common site for osteoporotic fractures. The other bone is the ulna. From this image it is clear that the 0.33 mm pixel size is just sufficient for the trabecular structure to be visible. Also the level of contrast is such that the cortical shell is quite distinct from the inner trabecular bone.

The analysis software provided by STRATEC to calculate density, separates trabecular bone from cortical and subcortical bone in the radius. This partition is achieved by an iterative contour detection starting at the inner cortical bone edge. All pixels within the contour are counted as trabecular bone while those outside the contour are assigned to the bone cortex. A factory installed calibration line is used to relate the measured linear attenuation coefficients comprising the image matrix to units of bone mineral density given in mg cm^{-3}. The formulation of this calibration line is as follows:

$$\left(\frac{mg}{cm^3}\right) = 982.723\,(LAC) - 224.0 \qquad 4.1$$

It is the grams of mineral which is obtained from the calibration. A volumetric density is obtained by dividing the grams of mineral by the product of the cross-sectional area and slice thickness.

In a given pQCT scan of the distal end of the radius four tissue components are present. They are fat, muscle, trabecular bone and cortical bone. The LAC of these four components at 40 kev are listed in table 4.1. The corresponding density derived from

Figure 4.4: A pQCT scan of the distal end of the left radius of a normal 23 year old female.

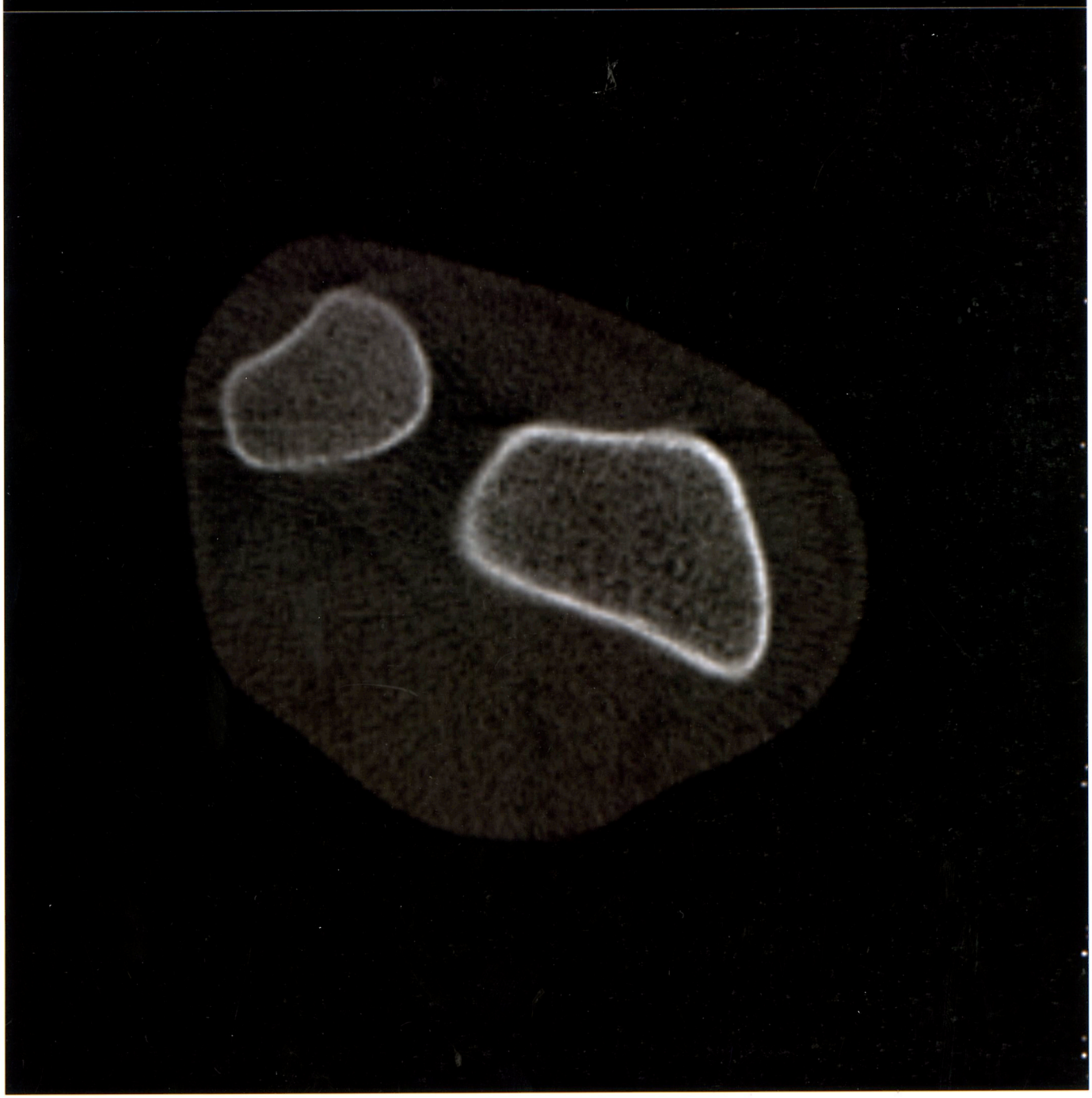


Table 4.1: Characteristics of tissues imaged by pQCT. The Linear attenuation coefficients (LAC) are determined at 40 keV and the pQCT density is determined from calibration.

Tissue	LAC (cm^{-1})	pQCT ($mg\ cm^{-3}$)
Fat	0.23	0
Muscle	0.30	60
Trabecular bone	0.50	260
Cortical bone	1.50	1200

calibration is also given. The zero density from fat results from the intercept term and ensures that the marrow fat integrated within the trabecular bone network does not contribute to the measured density of trabecular bone.

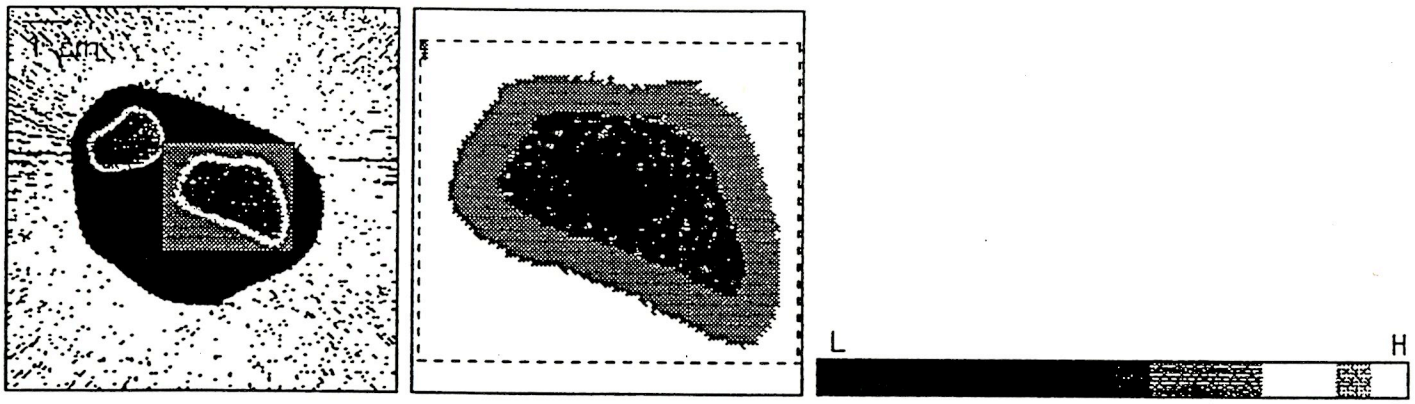
Figure 4.5 shows a typical report summary generated by the XCT-960X. This report corresponds to the pQCT image shown in figure 4.4. As noted, the result for trabecular bone is reported separately from cortical and subcortical bone. A true volumetric density of each bone component is obtained by dividing the grams of mineral by the product of the cross-sectional area and slice thickness.

4.3 Image Quality for Structural Assessment

The reconstructed pQCT image is essentially a map of energy weighted x-ray attenuation values in the scanned tissue slice. Its accuracy is constrained by intrinsic limitations of the system design. Dose constraints, and limits on x-ray tube output and detector efficiency, cause statistical uncertainties in the measured LAC's. Finite sampling of the image space imposes limits on the object spatial frequencies reproduced in the image. These discrepancies can affect how faithfully the trabecular bone network can be represented in the final reconstructed image. In the following sections, the effects of image noise and spatial frequency limits will be discussed with respect to the limitations each places on an assessment of trabecular bone structure at the distal end of the radius. The results of tests designed to quantify

STRATEC XCT-960X PQCT[tm]

Ct.Nr.: 0 Geschl.: W
Geb.: 13.02.71 Scandatum: 20.12.94 Alter: 23



	Gesamt	Spongiosa	Kort.(Sub)
Dichte:	407.6	215.9	564.7
[mg/ccm]	±5.0	±3.0	±9.0
#Voxels:	3271	1473	1798
≡[mm²]:	356.5	160.5	195.9

Figure 4.5 A summary report of the densities calculated from a 23 year old female (see figure 4.4). The total density is indicated by "Gesamt", the trabecular density by "Spongiosa", and the cortical and subcortical density by "Kort.". All density values are reported in mg/ccm.

the magnitude of image noise, subject contrast, and spatial frequency limits will be given.

4.3.1 Noise and spatial uniformity

Noise limits the perceptibility of low contrast detail in an image. From the image shown in figure 4.4, it is clear that much of the relevant detail of the trabecular structure is low contrast in nature. The ease with which this structural detail can be segmented and quantified is dependent on the difference in the average LAC's between two adjacent regions of the image. If a homogeneous medium is scanned, the variation in the linear attenuation about an average value is the noise of the imaging system. If all voxel values were the same, system noise would be zero. To assess the noise in a pQCT image a 5 cm diameter cylindrical water phantom was scanned. The noise level was defined as the standard deviation of a large group of voxels obtained from the image of the water phantom. A mean value of 293 cm^{-1} with a standard deviation (Sd) of 39 cm^{-1} was calculated from a circular region of interest (ROI). This Sd translates into a percent noise level of 13.3%. This level of noise was unchanged when evaluated five times over a three month period.

A measure of spatial uniformity is directly related to system noise. It is a measure of how constant the linear attenuation coefficient is in the centre and at the periphery of the reconstructed image of a homogeneous phantom. Testing of spatial uniformity was done again by scanning a 5 cm diameter cylindrical

water phantom. Five ROI's were defined in the reconstructed image, one at the centre and four on the periphery of the image. The mean and standard deviation of the LAC's in each of the five ROI's was calculated and compared. All means were within 1 Sd of each other. This agreement indicates that the pQCT imaging system exhibited acceptable spatial uniformity. Again, this degree of spatial uniformity was unchanged when evaluated five times over a three month period.

It is important to rationalize the measured noise level in the context of the limitations it may pose on extracting structural information from the pQCT image. This is done in the following examination of the subject contrast present at the distal end of the radius.

4.3.2 Subject contrast

On cross-section, the anatomy at the distal end of the radius is comprised of a soft tissue layer, a dense and heavily attenuating ring of compact bone, and a less dense network of trabecular bone and bone marrow. The ease with which the structural detail of the trabecular network can be segmented and quantified is dependent on the difference in the average LAC between two adjacent regions in the image. This difference is measured as the subject contrast (SC).

When testing the performance of CT scanners SC is defined relative to water and is expressed as:

$$SC=\frac{k}{\mu_w(E)}[\mu_1(E)-\mu_2(E)] \qquad 4.2$$

The terms μ_w(E), μ_1(E), μ_2(E) are the energy dependent linear attenuation coefficients of voxels containing water, material 1, and material 2, respectively. k is the CT number scaling constant (1000). Recalling that STRATEC has chosen to report voxel intensities as the linear attenuation coefficient scaled by 1000, then the definition of SC can be augmented to:

$$SC=1000[\mu_1(E)-\mu_2(E)] \qquad 4.3$$

For segmentation, the magnitude of SC is most important for μ_1(E) corresponding to trabecular bone and μ_2(E) representing bone marrow. Therefore, values of SC were calculated to determine the level of contrast between trabecular bone and fat and between cortical bone and fat. The LAC for fat, cortical bone, healthy trabecular bone and osteoporotic trabecular bone are listed in table 4.2. They are tabulated at 5 keV intervals and span an energy range of 25 to 60 keV. The LAC for fat and cortical bone were derived from the fourth order polynomials suggested by Webber (1987). Attenuation coefficients for trabecular bone were calculated using equation 4.4 and assuming that trabecular bone has the same composition as cortical bone.

$$\mu_t=\frac{\rho_t}{\rho_c}\mu_c+(1-\frac{\rho_t}{\rho_c})\mu_m \qquad 4.4$$

In this equation ρ_t and ρ_c represent the volumetric densities of trabecular bone and cortical bone and μ_t, μ_c, and μ_m represent the

Table 4.2: The linear attenuation coeffients (LAC) for fat, cortical bone, healthy trabecular bone and osteoporotic trabecular bone.

Energy (keV)	<---------- LAC (cm^{-1}) ---------->			
			(Trabecular bone)	
	Fat	Cortical	Healthy	Osteoporotic
25	0.3040	3.6751	0.9783	0.4726
30	0.2705	2.1515	0.6467	0.3645
35	0.2467	1.4717	0.4917	0.3080
40	0.2291	1.1135	0.4060	0.2733
45	0.2155	0.9011	0.3526	0.2498
50	0.2048	0.7632	0.3164	0.2327
55	0.1961	0.6669	0.2903	0.2196
60	0.1890	0.5952	0.2702	0.2093

LAC for trabecular bone, cortical bone, and marrow fat, respectively. The ratio of the volumetric densities (ρ_t/ρ_c) was taken to be 0.2 for healthy trabecular bone. This fraction was reduced to 0.05 for osteoporotic trabecular bone. It is expected that the attenuation coefficients of most subjects will fall between these two extremes. The magnitude of SC for cortical bone and trabecular bone relative to fat were derived from the LAC tabulated in 4.2. The results are plotted in figures 4.6 and 4.7. In both plots it is not surprising that the magnitude of SC increases with decreasing energy. At 40 keV the contrast difference between fat and cortical bone is approximately five times that of the contrast difference between fat and healthy trabecular bone. Most importantly, two points are worth highlighting with respect to the contrast differences for the two states of trabecular bone. First, at 40 keV both healthy and osteoporotic trabecular bone are segmentable from fatty marrow. This point is illustrated by the fact that the values of SC are greater than zero. Second, the contrast difference for severely osteoporotic trabecular bone is 44. With the 13% image noise level derived in section 4.3.1, the voxel intensities for fat are expected to vary with a standard deviation of approximately 29. The magnitude of SC for osteoporotic trabecular bone at 40 keV is 44 and is approximately 1.5 standard deviations above the expected image noise level. Therefore, even at this extreme diseased state, the trabecular bone in a pQCT image can be segmented from marrow.

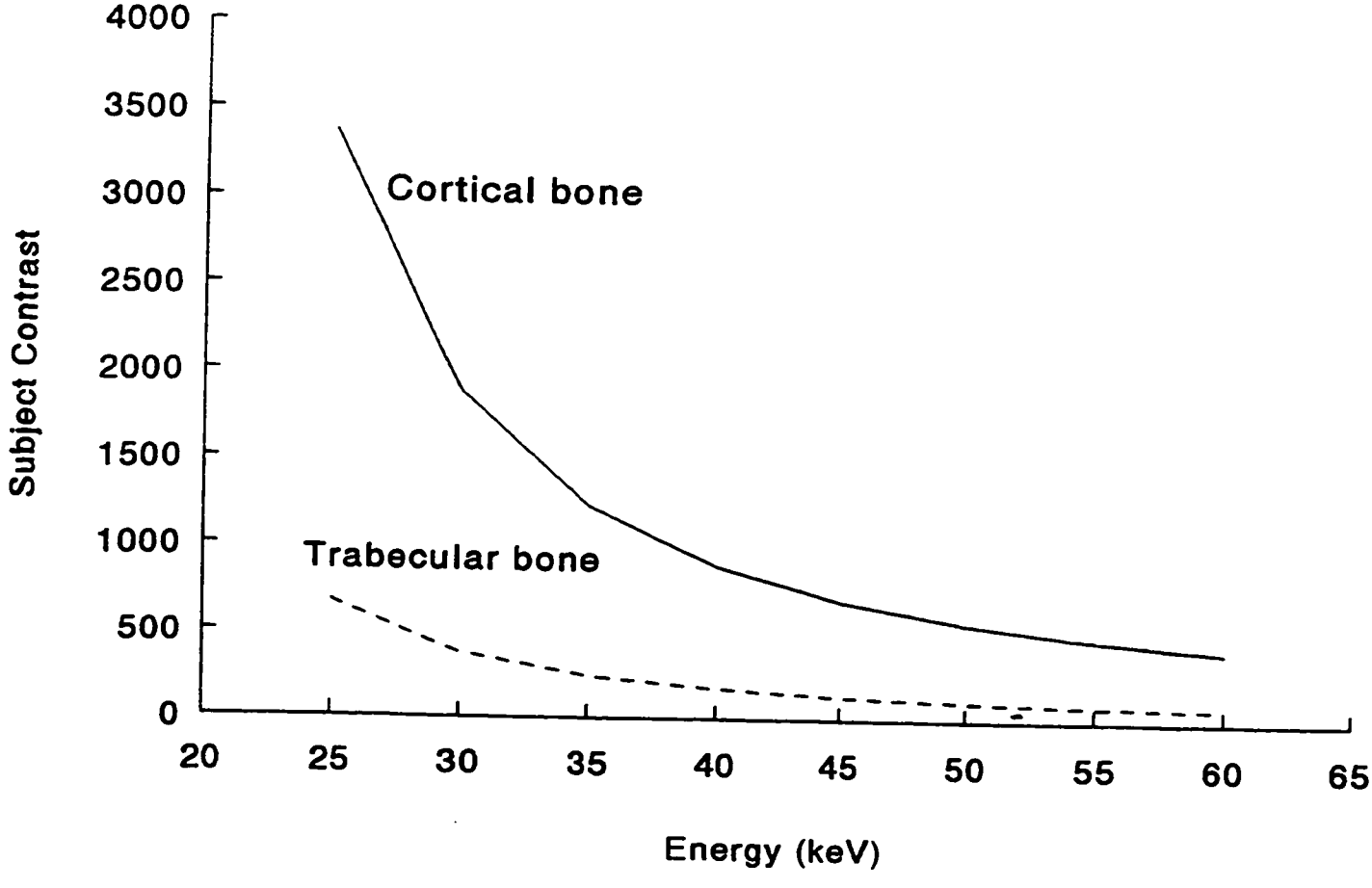


Figure 4.6 A comparison between the subject contrast (relative to fat) for normal cortical bone and trabecular bone for energies between 25 to 60 keV. At 40 kev there is almost an order of magnitude difference between the contrast levels of the two bone types.

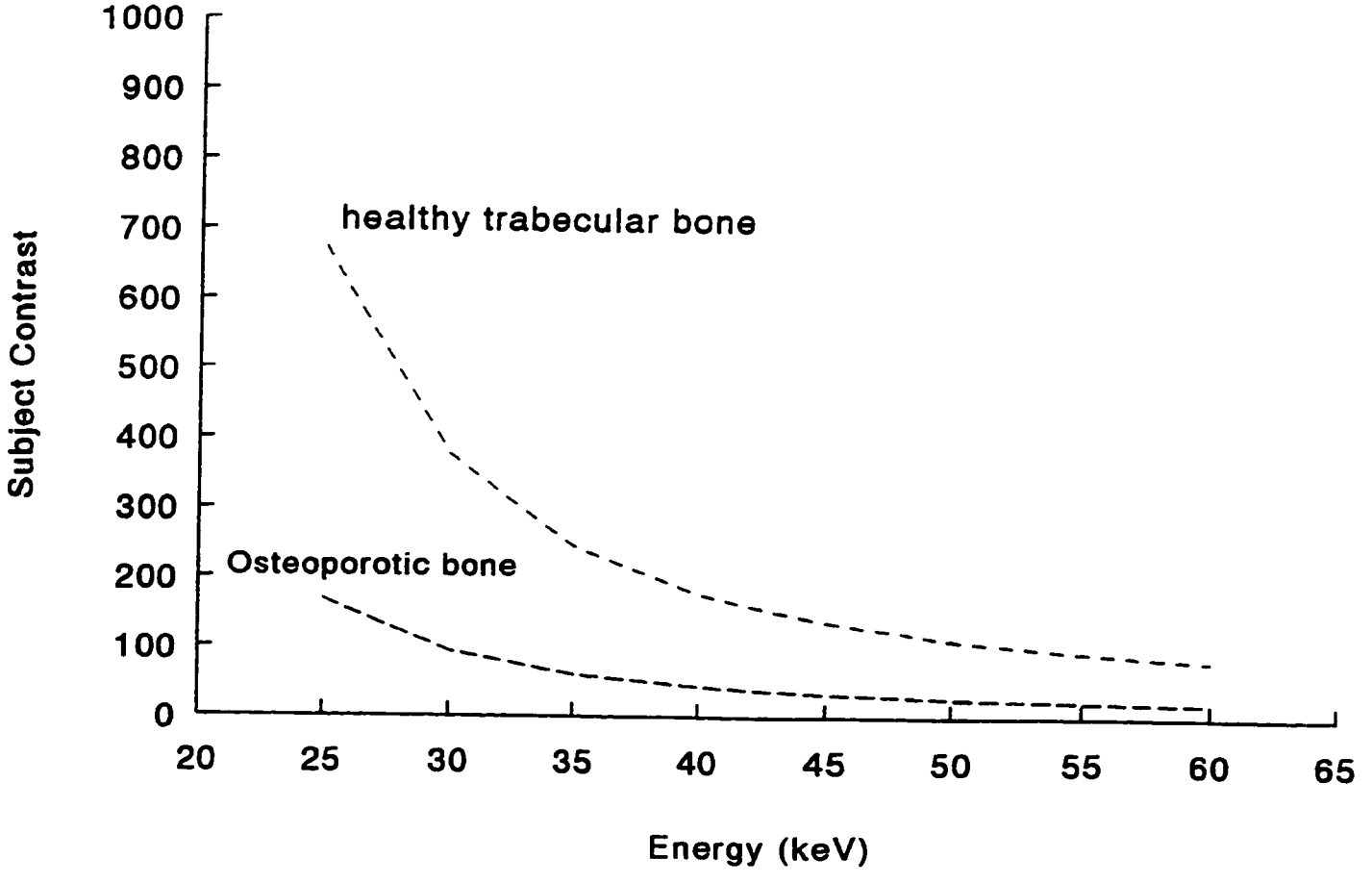


Figure 4.7 A comparison between the subject contrast (relative to fat) for normal trabecular bone and one that is representative of osteoporosis. At 40 kev there is a factor of four difference between the contrast levels of the two bone states. The contrast level of the diseased bone is greater than zero which suggests it can be differentiated from a soft tissue background.

4.3.3 Line spread function

As previously stated the x-ray beam has a mean energy of 40 keV. The finite size of the x-ray beam leads to degradation of the projection data. These degradations blur the edges of structures in the final reconstructed image (Verly and Bracewell 1979). To estimate the magnitude of this blurring the line spread function (LSF) of the scanner was evaluated by imaging a thin strip of aluminum foil in air. The foil strip was placed at the centre of the scanner with its long axis perpendicular to the image plane and scanned five times. To reduce the effects of statistical fluctuations, the LSF was determined from the average of these five scans. Figure 4.8 shows a line profile recorded across the averaged image of the strip of aluminum foil. The data points plotted represent the linear attenuation coefficients along the profile drawn normalized to the maximum value in the image. The data points are fitted to an analytical equation representing the weighted sum of an exponential and gaussian function as described by Boone and Siebert (1994). This fitted profile represents the line spread function. A full-width half-maximum (FWHM) of 0.49 mm was determined from the fit coefficients. This measured FWHM is significantly larger than the average size of a trabeculae (0.2 mm) but significantly smaller than the average size of a marrow pore (0.75 mm). This suggests that at the 0.33 mm voxel size, the spatial extent of the marrow space within the trabecular network will be reproduced well. However, this will be somewhat offset by the blurring of individual trabeculae.

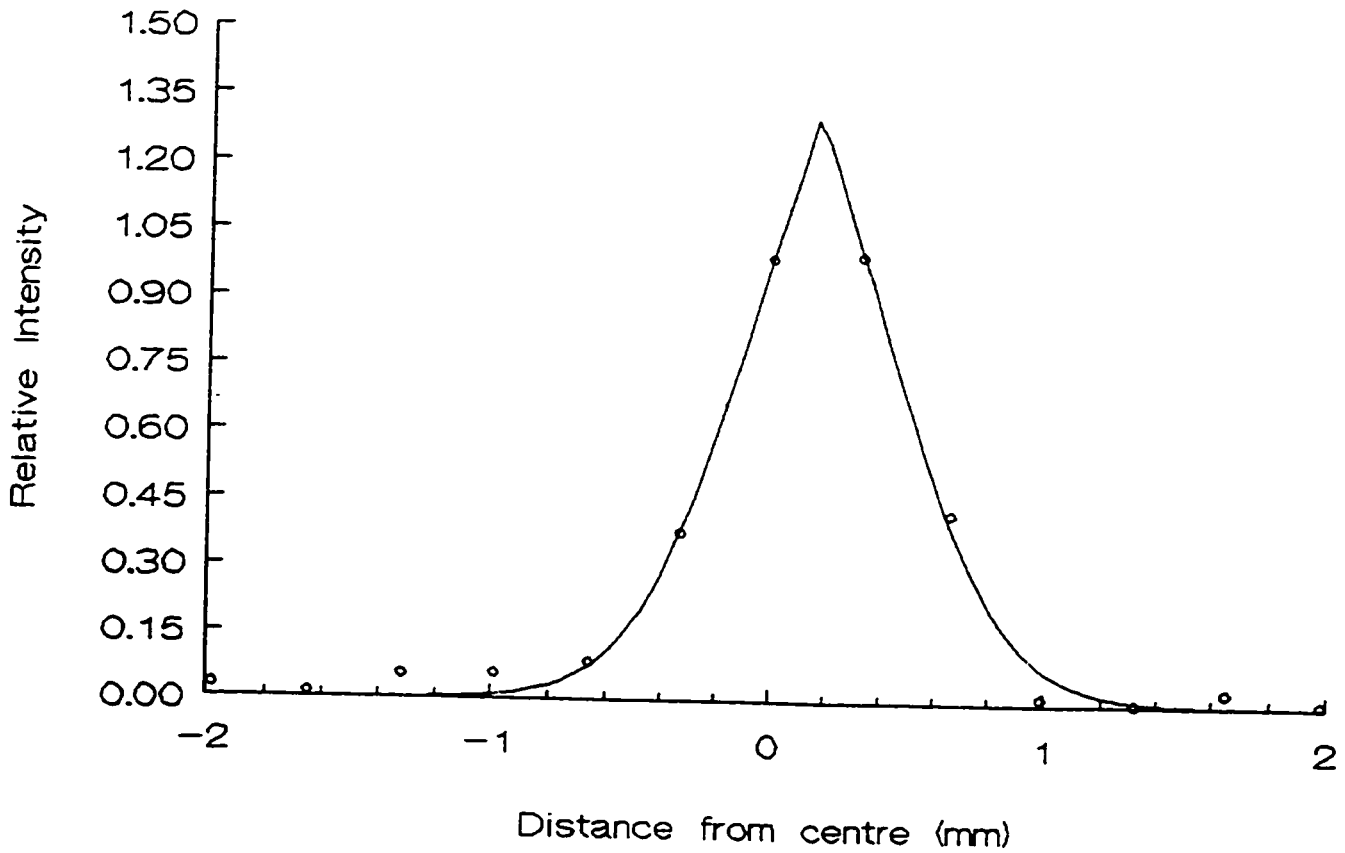


Figure 4.8 The linear attenuation profile across a thin strip of aluminum foil used to assess the LSF of the pQCT scanner. Data points (o) were normalized to the maximum linear attenuation coefficient present in the image.

4.3.4 Linearity

As previously stated, it is the linear attenuation coefficient that identifies the type of tissue in the image and provides a level of contrast between different tissues. Therefore, an important check of scanner performance is the ability to reliably determine the linear attenuation coefficients of a set of materials of known density and x-ray absorption properties. As a check of the XCT 960's performance, a set of cylindrical rods of different material with known physical and x-ray absorption properties were imaged. The different materials along with their physical properties are shown in table 4.3. Following a scan of each material, the mean and standard deviation of the linear attenuation coefficient was determined from a ROI set within each image. The measured attenuation values were compared against the theoretical values listed in table 4.3. The results of this comparison are plotted in figure 4.9. The straight line represents a weighted least squares fit to the data. The slope, intercept, and correlation coefficient are indicated on the plot. The high correlation coefficient indicates that the scanner behaves linearly. The fact that the slope and intercept are not significantly different from one and zero respectively, indicates that the scanner correctly measures the linear attenuation coefficients of the materials of interest at 40 keV.

4.3.5 Beam hardening

The transmission equations used to derive the attenuation

Table 4.3: Characteristics of the materials used to check scanner linearity. The linear attenuation coefficients are determined at 40 keV.

Material		Density ($g\ cm^{-3}$)	Linear attenuation coefficient (cm^{-1})
Polypropylene	C_3H_6	0.85	0.198
Polyethylene	C_2H_4	0.94	0.210
Water	H_2O	1.00	0.282
Lucite	$C_5H_8O_2$	1.19	0.290
Lexan	$C_{16}H_{14}O$	1.20	0.272
Hydroxyapatite phantom	$Ca_5OH(PO_4)_3$	0.38	0.494

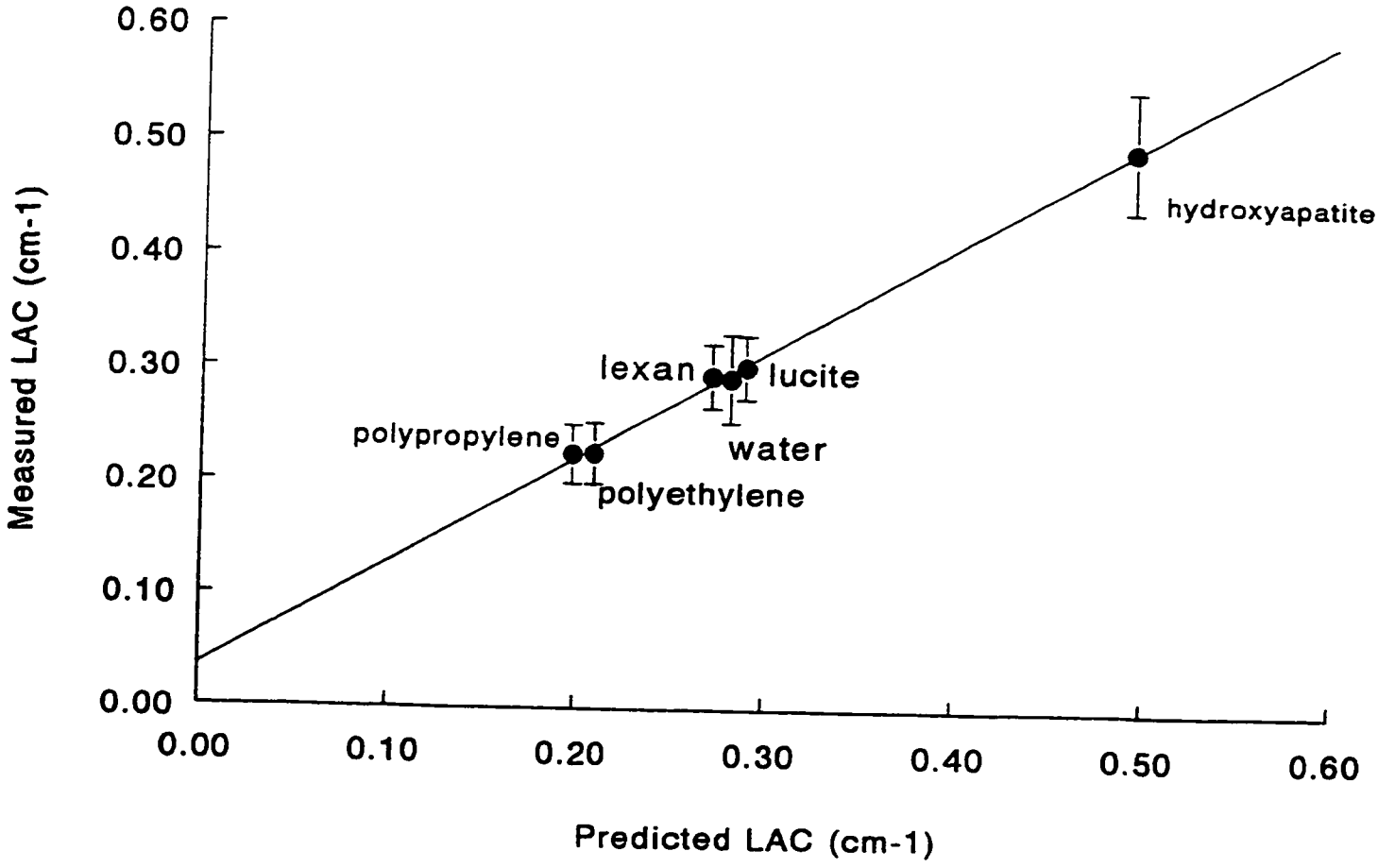


Figure 4.9 A check of scanner linearity. The points plotted represent the mean LAC values derived from a region of interest set within the image of each material scanned. The error bars represent the standard deviation associated with the mean. The fitted line represents a weighted least squares fit to the data points. The fitted slope and intercept are (0.93±0.08) and (0.04±0.05), respectively. The correlation coefficient is 0.997.

coefficients in a pQCT image assume that the radiation used to scan the object or subject is monoenergetic. However, the x-ray beam used is polyenergetic. More specifically, in the XCT 960 scanner the x-ray tube operates at 45 kVp and produces a beam with an effective energy of 40 keV. The full width at half maximum of the beam after 5 mm of aluminum filtration is 8 keV. So, as this heterogeneous beam passes through the object being scanned the lower energy photons are removed preferentially and the beam becomes harder with depth. For a given x-ray spectrum, the degree of beam hardening depends on the composition and diameter of the object being scanned. The distal end of the radius is comprised of a soft tissue layer surrounding a heavily attenuating ring of compact bone which encompasses a less dense mixture of trabecular bone and marrow. If not corrected for the soft tissue layer, the cortical shell will be characterized by lower energy photons than the centre of the bone. This results in voxels near the centre of the radius being assigned lower attenuation coefficients than voxels near the periphery. The linear attenuation coefficients throughout the image are directly related to density. Therefore, correct LAC must be determined if the correct bone density distribution is to be recorded. To ensure this accuracy, the transmission data acquired during a scan are corrected for beam hardening using a fourth order polynomial.

The severity of the beam hardening effect can be characterized by plotting the values of the LAC across the diameter of the object scanned. The degree to which the resulting profile is concave at

the centre of the object indicates the severity of beam hardening. As a check, the attenuation coefficient profile along the diameter of the manufacturer's bone mineral equivalent phantom was determined. A cross-sectional image of this phantom, along with the profile along its diameter, is plotted in figure 4.10. The phantom is comprised of a cylindrical ring of soft tissue equivalent material with an outer diameter of 5.5 cm, a ring of bone equivalent material to simulate compact bone and a less dense mixture at the centre to simulate trabecular bone. Clearly, there is no visible cupping effect present at the centre of the bone mineral phantom. This confirms that beam hardening effects are negligible and do not affect a determination of density nor does it bias an assessment of structure.

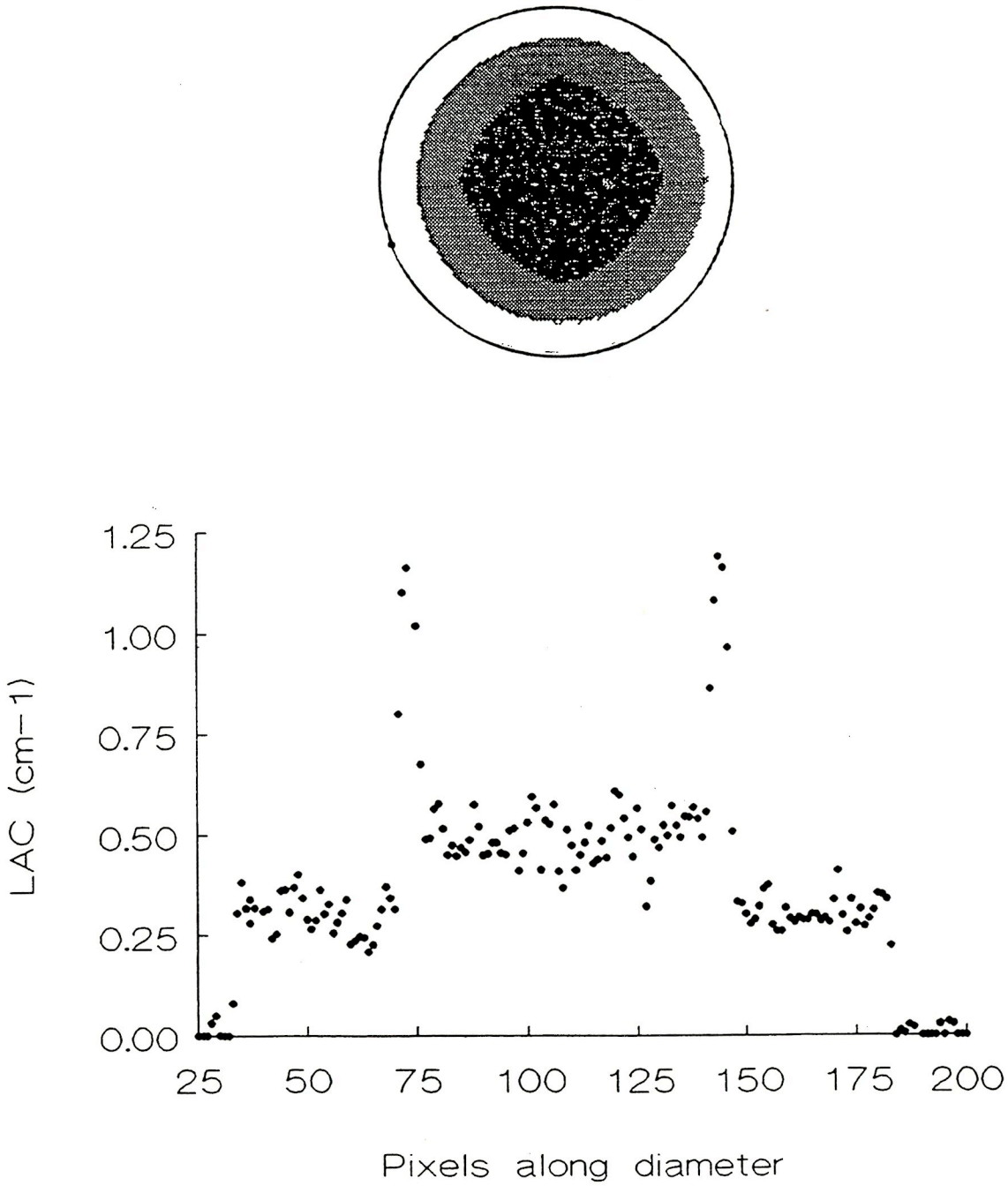


Figure 4.10 The linear attenuation coefficient profile across the diameter of the calibration phantom. The image of the phantom has been enlarged to reveal a soft tissue equivalent layer, a compact bone shell, and a trabecular bone equivalent centre. Note that these three tissue components are clearly present in the profile.

Chapter 5

pQCT AND MR IMAGE SEGMENTATION

5.0 Introduction

Previous work to characterize trabecular structure at the distal radius applied image analysis techniques to plain film radiographs (Rockoff et al 1971, Geraets et al 1990). In radiographic imaging the three dimensional structure within the imaged bone is summed and represented by a two dimensional image. These effects of volume averaging blur the bone structures in the final image limiting the degree to which the architecture of the bone can be characterized. These limitations can be overcome by acquiring cross-sectional images of sufficient resolution and slice thickness. Two studies have demonstrated the feasibility of extracting structural information from pQCT images. First, by using run-length analysis, a close relationship between histomorphometric values and run-length parameters was demonstrated from images of the distal radius and tibia (Durand and Ruegsegger 1991). However, direct interpretation of the run-length parameters as indicators of structure required simulating two and three-dimensional models of the trabecular architecture. In the second study a two-dimensional representation of the topology of the trabecular bone at the ultra-distal radius was formed by stacking a series of contiguous slices. Analysis of the structural pattern in the resulting topology revealed differences between a post menopausal and a normal population (Takagi et al 1995). No differences were detected between the two groups with a measure of trabecular or cortical bone density.

More direct and computationally efficient methods can be

applied to assess bone quality from high resolution pQCT and MR images. With this in mind, this chapter presents the details of an algorithm to segment the trabecular network at the distal end of the radius based primarily on a region grow and skeletonization step. The implementation of the postprocessing algorithm is described and various indexes of structure are suggested all of which can be derived from the processed image.

5.1 pQCT Image Segmentation

After evaluating trabecular and cortical bone density, the pQCT images are transferred from the XCT 960 to a Sun workstation (Sun Microsystems, California) running the MUMC DISPLAY package developed in the Department of Radiology at the McMaster University Medical Centre. The images are converted from the pQCT data file format into a form readable by MUMC DISPLAY. The C code for this conversion program is given in appendix B. The pQCT images are then processed using an algorithm written and implemented to derive structural information. The C code for this postprocessing algorithm is given in Appendix C. Our objective in postprocessing the pQCT image is to segment the trabecular bone from the original cross-sectional image and represent its structure by a simplified image from which various indices expressing its mechanical competence can be extracted. The only intervention required by the user during this analysis is the placement of a rectangular area of interest around the radius. Figure 5.1 shows the placement of a typical area of interest set around the radius of a 23 year old female volunteer. Once this area of interest is selected the algorithm proceeds automatically

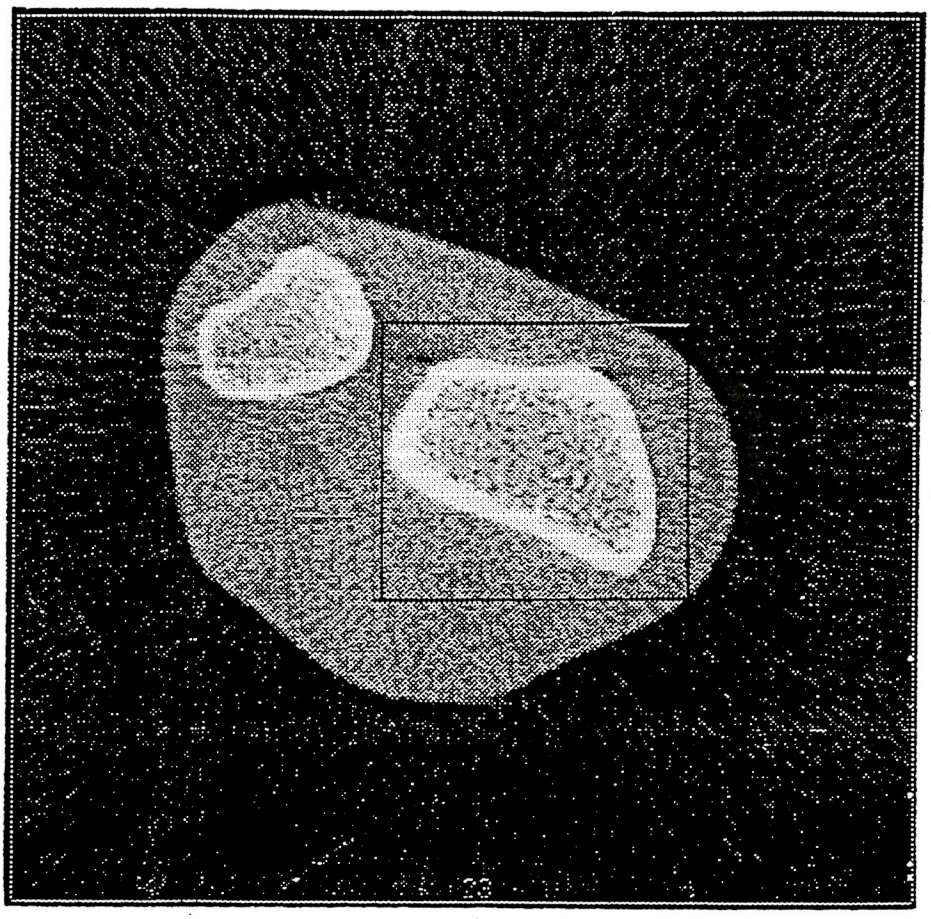


Figure 5.1: A typical region of interest set around the radius of a 23 year old female volunteer.

through the following steps.

5.1.1 Region growing

Applying a global threshold to segment bone from the surrounding soft tissue and the fatty marrow within the intertrabecular spaces, produces an incorrect representation of the bone structure. In the soft tissue region of the image, the spatial uniformity is such that a global threshold identifies several pixels outside the cortical shell as trabecular bone. Although these pixels may be removed by applying a median filter or a 3-point smoothing function before thresholding, this has the unwanted effect of blurring the structures in the trabecular network. Consequently, we chose to apply a region grow technique (Gonzalez and Woods 1992). As implied by the name, region growing groups pixels of similar properties into a larger connected region. The procedure requires selecting a "seed" pixel within the area of interest and applying a set of rules to govern which pixels are added to the seed to form the region. In our case the seed is taken as the pixel corresponding to the maximum linear attenuation coefficient in the rectangular area set around the radius. This maximum value occurs in cortical bone. Pixels are then grown from this seed using two rules. First, a candidate pixel is considered for addition to the region if it is 8-connected to at least one pixel already in the region. The scheme for 8-connectivity used by the region grow algorithm is outlined below.

$$\begin{matrix} n_8 & n_1 & n_2 \\ n_7 & P & n_3 \\ n_6 & n_5 & n_4 \end{matrix}$$

As shown, if pixel P is the original seed pixel, then candidate pixels which are 8-connected are given by n_1 through n_8. Alternatively, if P is part of the region already grown, at least one of n_1 to n_8 would already be in the region.

The second rule for pixel addition to the region is that the grey value of the candidate pixel should exceed a defined threshold. The selection of this threshold is critical because it has a direct bearing on the size and grey level composition of the region grown. To illustrate what features are considered in making this threshold selection, a histogram for the area set around the radius illustrated in figure 5.1, is shown in figure 5.2. The appearance of this histogram is typical of those encountered in most subjects scanned. It is, to a first approximation, a trimodal gaussian distribution. The most prominent peak on the left corresponds to the soft tissue background. The two other peaks, as labelled, correspond to trabecular bone and cortical bone. Due to the presence of subcortical bone and partial volume effects there is not a clear distinction between the two bone peaks. Given the consistency in the shape of the histogram determined from different subjects, a fixed threshold value was implemented to separate soft tissue and marrow from bone. This value was determined as follows. An area of interest containing only soft tissue was defined in ten subjects. The linear attenuation coefficient corresponding to two standard deviations above the mean linear attenuation coefficient in the soft tissue region defined was obtained from each of the ten subjects. These ten values were then averaged and considered to be the fixed threshold for all images. The coefficient of

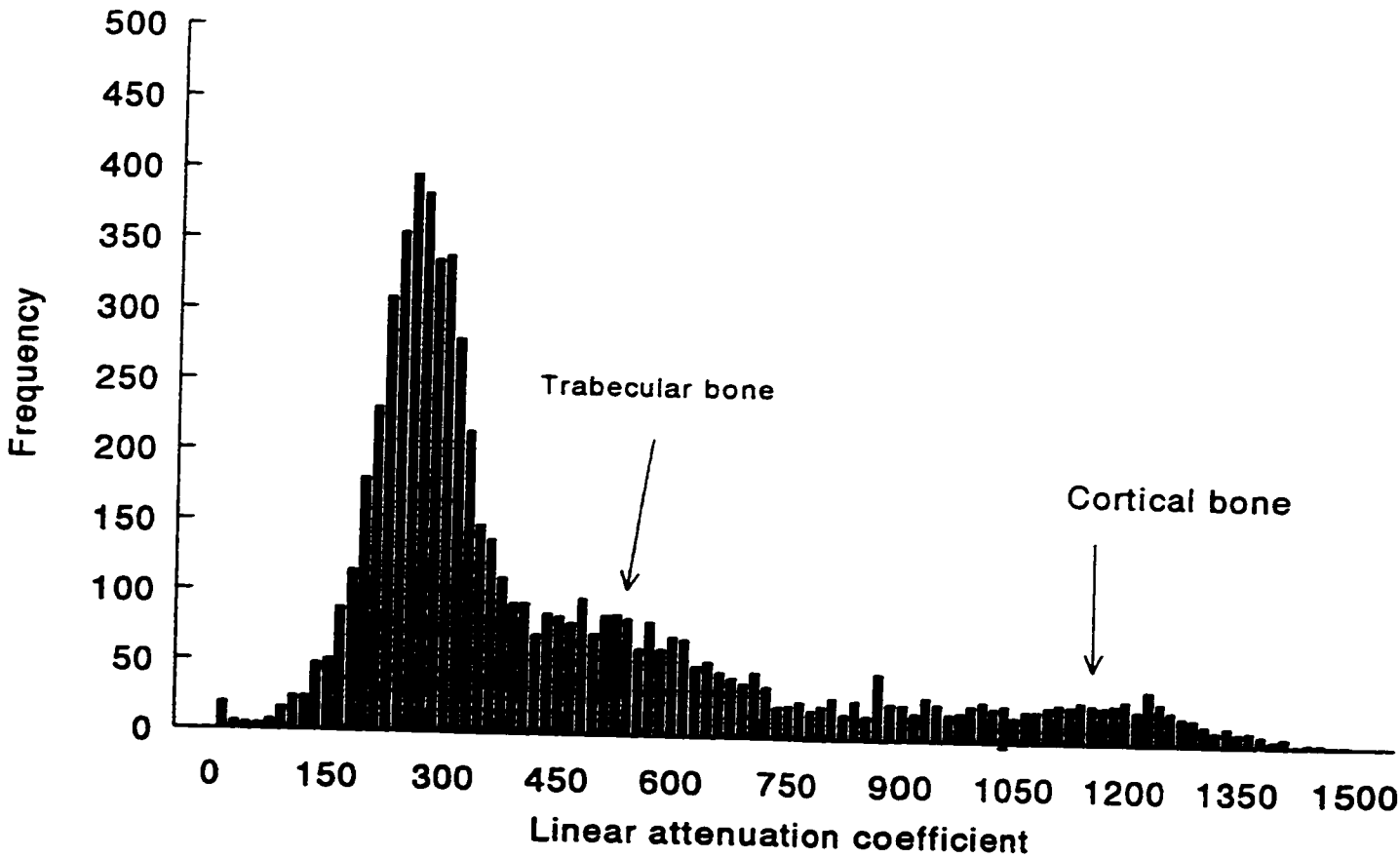


Figure 5.2: A typical histogram of the linear attenuation coefficients($x10^{-3}$) present in an area of interest defined around the radius.

variation of this average threshold was 2.3%.

The region grow algorithm terminates when the current size of the region grown remains unchanged. Upon termination all pixels within the region represent a connected bone area. All pixels in this bone area are labelled with their original grey value while those in marrow and outside bone are labelled as background. An example of the effectiveness of the region grow technique is shown in figure 5.3b. The few bone pixels grown beyond the cortical shell are removed by defining the outer contour of the bone using a line by line scanning algorithm. All pixels within the contour are kept and considered as being part of the bone structure to be processed further. We also identify those pixels, within the contour, excluded by region grow because they fail the test for connectivity but have a pixel value above the threshold for trabecular bone. The identification of such pixels ensures that "islands of bone" which appear isolated on cross-section are included in the analysis because they may be connected in the third dimension along the long axis of the bone. These isolated fragments of bone also appear in two-dimensional axial sections prepared for biopsy studies of trabecular structure (Mosekilde 1988).

5.1.2 Binary representation of bone structure

The grey level image which results from region grow is converted into a binary image using three standard image processing steps. First, to better visualize the trabecular structure at the radius, the image is subjected to a histogram equalization technique. This increases the dynamic range of the

pixels present in the image allowing better delineation of trabecular, subcortical and cortical bone pixels. Second, the histogram equalized image is sharpened by applying a high pass spatial filter based on the 3x3 Laplacian mask. This mask is given below.

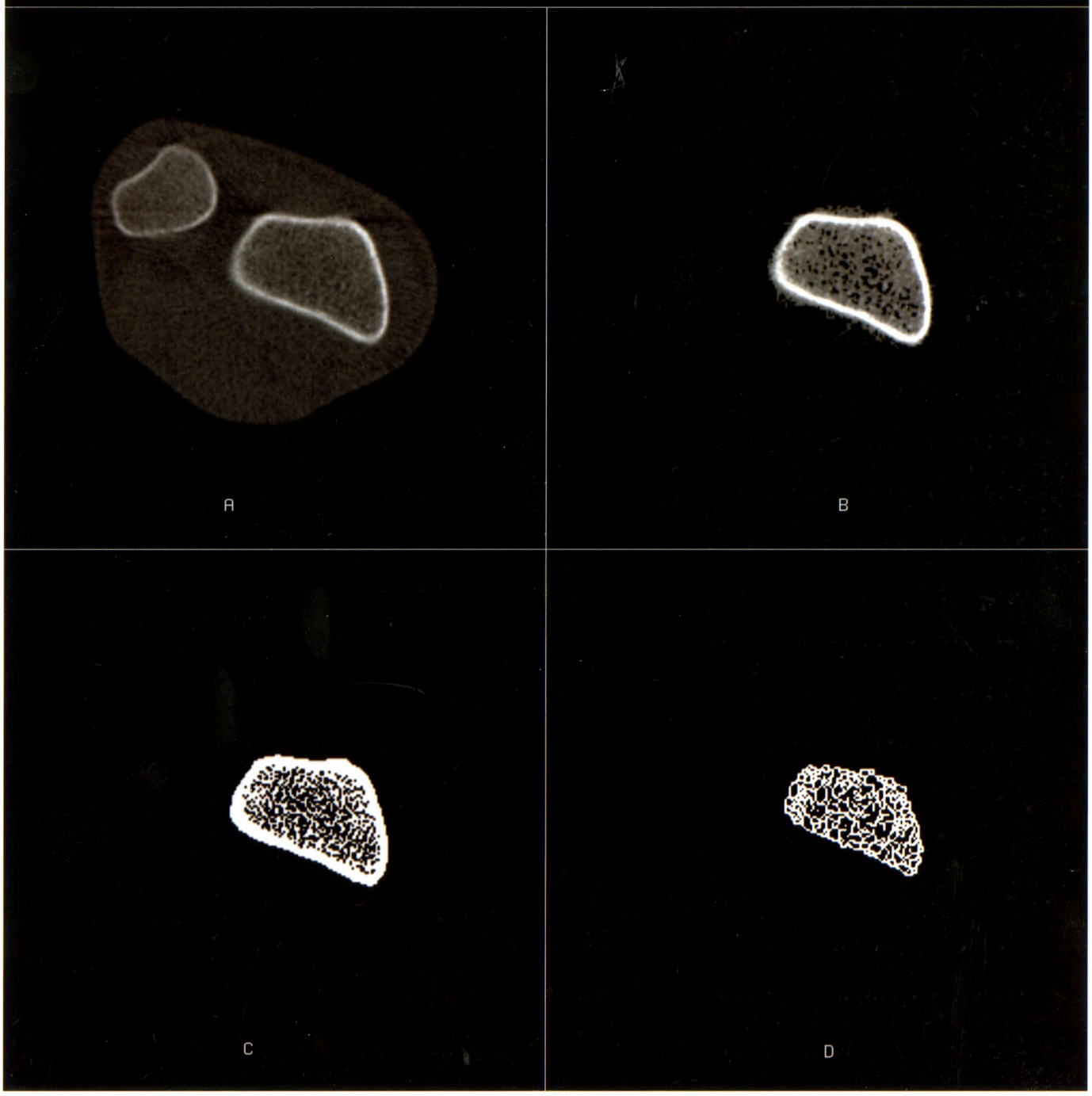
$$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$$

The principle objective of sharpening the image is to highlight the fine details in the structure of the trabecular network that have been blurred during the image acquisition process. Third, the final binary representation of the bone distribution is created by applying a global threshold to the sharpened image. The selection of the threshold is simple after the application of the sharpening mask because the histogram of the resulting image has a clear bimodal distribution (Gonzalez and Woods 1992) which can be seen in figure 5.4 which shows the histogram of the sharpened image. A typical binary representation of the distribution of bone at the radius is shown in figure 5.3c. As shown, the bone distribution in the original image is accurately reproduced. Also, the trabecular network appears uniform and its shape real.

5.1.3 Skeletonization

An important approach to representing the structural shape of an object recorded in an image is to reduce it to a graphical representation (Gonzalez and Woods 1992). This graphical representation may be produced by obtaining the skeleton of the object via a thinning algorithm. Skeletonization or thinning

Figure 5.3: The postprocessing steps used to assess trabecular bone structure at the distal radius. The original cross-sectional image (A) is of a 23 year old female with trabecular density 215.9 mg/cc. The bone structure at the radius is segmented by region grow (B), represented as a binary image (C), and thinned to produce a representation of trabecular bone from which connectivity can be assessed (D).



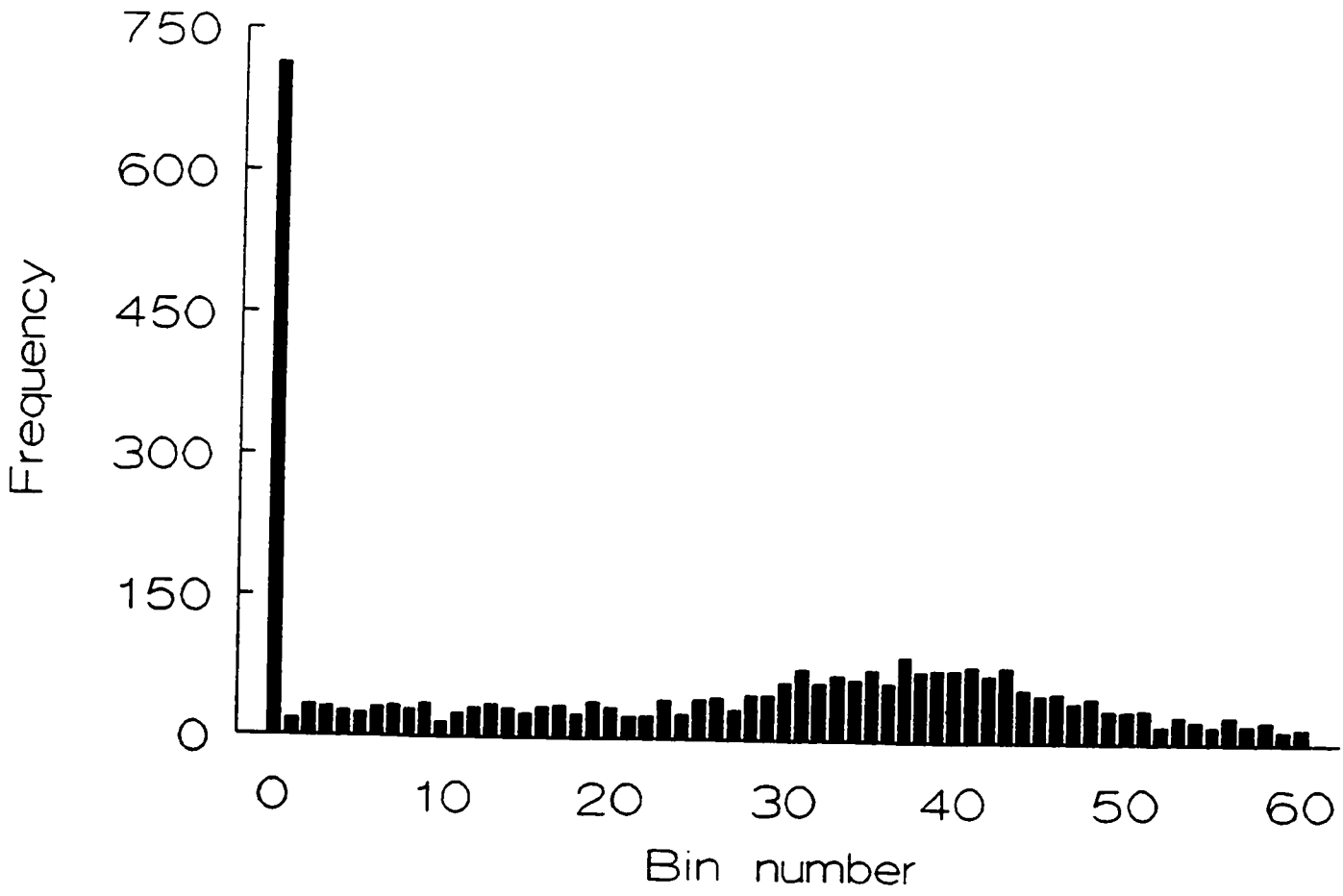


Figure 5.4: Resulting histogram after edge enhancement with a sharpening filter. Note the clear bimodal distribution with the first peak at bin 0 and the second peak at bin 40.

routines erode a binary image until a centre line of 1 pixel thickness remains. We chose to implement the parallel thinning algorithm developed by Zhang and Suen (1984) which produces accurate and connected representations of a range of binary shapes. The algorithm reaches a final skeleton by repeatedly peeling off the border of the object in a sequence of passes through the image. The connectivity of the final skeleton is ensured by defining the peeling conditions so that no pixel is removed that breaks local connectivity. Before thinning the binary image, the inner contour of the cortical shell is defined using an automatic contour detection algorithm which makes use of compass gradient masks (Robinson 1977). All pixels beyond the contour are removed and the remaining trabecular network is thinned. The results of applying the algorithm are shown in figure 5.3d. This skeleton is a visually pleasing representation of the trabecular network present in the original image. The inner contour of the cortical shell remains to ensure connectivity. From this skeleton various indices of trabecular connectivity can be extracted.

5.2 Indices of Structure

The following sections describe a proposed set of indices which can be derived from the processed pQCT image. These indices quantify the trabecular architecture at the distal radius by measuring network connectivity and inter-trabecular spacing. From this information, the mechanical competence of the bone architecture can be inferred.

5.2.1 Network connectivity

Strut analysis was applied to quantify the degree of connectivity of the bone architecture represented in the skeleton image (Compston et al 1993). In strut analysis the network examined is considered to consist of a number of one-dimensional struts. This treatment of the trabecular bone network is sketched in figure 5.5. As sketched, the junction between three or more struts is defined as a node (Nd). A strut that is connected at one end and is free at the other end is labelled a free end (Fe). Those struts representing trabeculae which run perpendicular to the image plane appear as a point in the skeleton image. These are counted as isolated points (Ip). The total length of the trabecular network is quantified as a network length (Nl). A well connected bone is characterized by a large number of nodes and few free ends.

A number of interesting uses have been made of these strut analysis parameters either in combination or as single indicators of connectivity. For example, to assess connectivity in iliac crest biopsy samples acquired from patients suffering from primary hyperparathyroidism, the strut parameters were normalized to the total bone area and compared (Parisien et al 1992). Alternatively, others have defined a trabecular fragmentation index from in-vivo CT images of vertebrae based on the the number of discontinuities per unit length (Chevalier et al 1992). Discontinuities were scored as isolated points and free ends. However, because of how these vertebral images were processed, the point at which individual trabeculae attach to the cortical shell are also scored, perhaps incorrectly, as free ends.

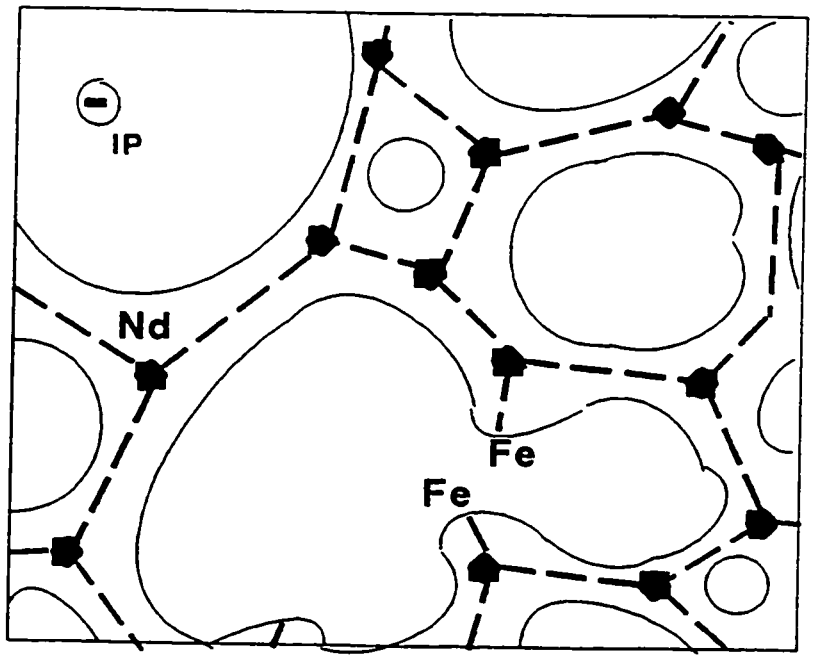


Figure 5.5 Trabecular strut analysis of the two-dimensional trabecular bone structure. The bone network is represented by a series of one-dimensional struts shown here as broken lines. Junctions in the network are indicated by nodes (Nd) and discontinuities by free ends (Fe) and isolated points (Ip).

We chose to implement a variation on the trabecular fragmentation factor suggested by Chevalier and others (1992). We combine Nd, Fe, Ip, and Nl into a single "connectivity index" (CI) to maximize the information used to quantify structural integrity. We define CI as follows:

$$CI=\frac{(Nd - Fe - Ip)}{Nl} * 100 \qquad 5.1$$

With this formula we weight the mechanical importance of each index in the numerator equally. That is, the positive effect of one node on mechanical stability will be cancelled out by one free end or isolated point. More importantly, we also make use of the fact that a change in connectivity such as a break along a strut will increase the number of free ends by two. Therefore, as defined, a large CI value reflects a high degree of connectivity while a low (perhaps negative) value reflects a weak and highly disrupted network. Division by the network length is intended to account for bone size. Conceptually the division by network length is required because if two structures had the same number of nodes, free ends, and isolated points but differed in network length, then the structure with the longer network length would be weaker and should have a lower CI.

5.2.2 Marrow pore size

An examination of the binary representation of the bone cross-section (figure 5.3c) reveals holes of various sizes. As shown histologically, the total number and area of holes in the trabecular network can give a clue about the structural competence of the bone network (Vesterby et al 1989b). To locate

the holes, the background in the binary image was considered to be a connected region which could therefore be marked using region grow. With this in mind holes are scored as follows. From a background seed the algorithm "grows" a hole by marking each background pixel that is 8-connected. The assignment of pixels to a hole is stopped when all possible paths of "growth" of that hole are obscured by bone. Another background seed pixel is selected and the procedure is repeated. The algorithm stops when all background pixels have been assigned to a hole. Upon termination the following indexes are determined. The number of regions grown represent the number of holes in the bone cross-section. The area of each hole grown is recorded allowing for the computation of a mean hole size H_A and a maximum hole size H_M.

5.3. MR Image Segmentation

All images were transferred from the 1.5 Tesla General Electric Signa clinical imager to a Sun Workstation(Sun Microsystems, Mountain View California) for processing. The objective in postprocessing the high resolution MR images was to segment the trabecular bone from the bone marrow and soft tissue background, and to represent its structure by a simplified image from which various indices expressing its mechanical competence could be extracted. The C code for the postprocessing algorithm is given in appendix D. Again, during the segmentation process the only intervention required by the operator was the placement of a rectangular area of interest around the radius. Once selected, the trabecular bone network was extracted in two stages. First, the boundary between cortical bone and trabecular

bone was defined using an automatic contour detection algorithm which makes use of compass gradient masks (Robinson 1977). Those pixels beyond the contour (cortical bone, muscle, fat) are excluded from further analysis. All remaining pixels within the contour correspond to trabecular bone and marrow. Figure 5.6b shows the effectiveness of the contour algorithm. In the second stage of segmentation, trabecular bone was separated from marrow. This is a critical step because the reliability of the various indices of structure depend on how accurately trabecular bone is separated from bone marrow. There are a number of thresholding techniques which may be applied to segment grey level images (Sahoo et al 1988). The selection of a given thresholding method is often governed by the level of noise and the degree of contrast present in the image. In an MR image of the trabecular structure at the distal end of the radius, the bone appears as a low intensity signal while the fat within the inter-trabecular space appears as a high intensity signal. An adaptive threshold was chosen to classify trabecular bone from fat. This thresholding scheme looks for variations in intensity to distinguish bone from fat. It does so by comparing the original image against a low pass version of itself (Gonzalez and Woods 1992). However, the adaptive threshold is sensitive enough to identify small intensity variations in marrow as trabecular bone. Therefore a second threshold was applied to eliminate those pixels with a signal intensity consistent with marrow but identified as part of the trabecular bone network. This threshold was set at 50% of the maximum value in the region of interest defined. Because there is very little overlying tissue covering

the distal end of the radius, this maximum value varied less than 10% as the depth of the image slice increased from the receiver coil. Also, although rare in occurrence, care was taken to exclude the very high intensity signals from vascular structures which may appear in the region. After thresholding, a binary representation of the trabecular structure at the radius is obtained. Figure 5.6c shows an example of this binary representation. As shown, the trabecular bone distribution in the original image is reproduced well.

A final representation of the structural shape of the trabecular network was obtained by applying a thinning algorithm to the binary image. As was done for the pQCT images, we chose to implement the parallel thinning algorithm developed by Zhang and Suen (1984) which produces connected representations of a range of binary images. The results of applying the algorithm are shown in figure 5.6d.

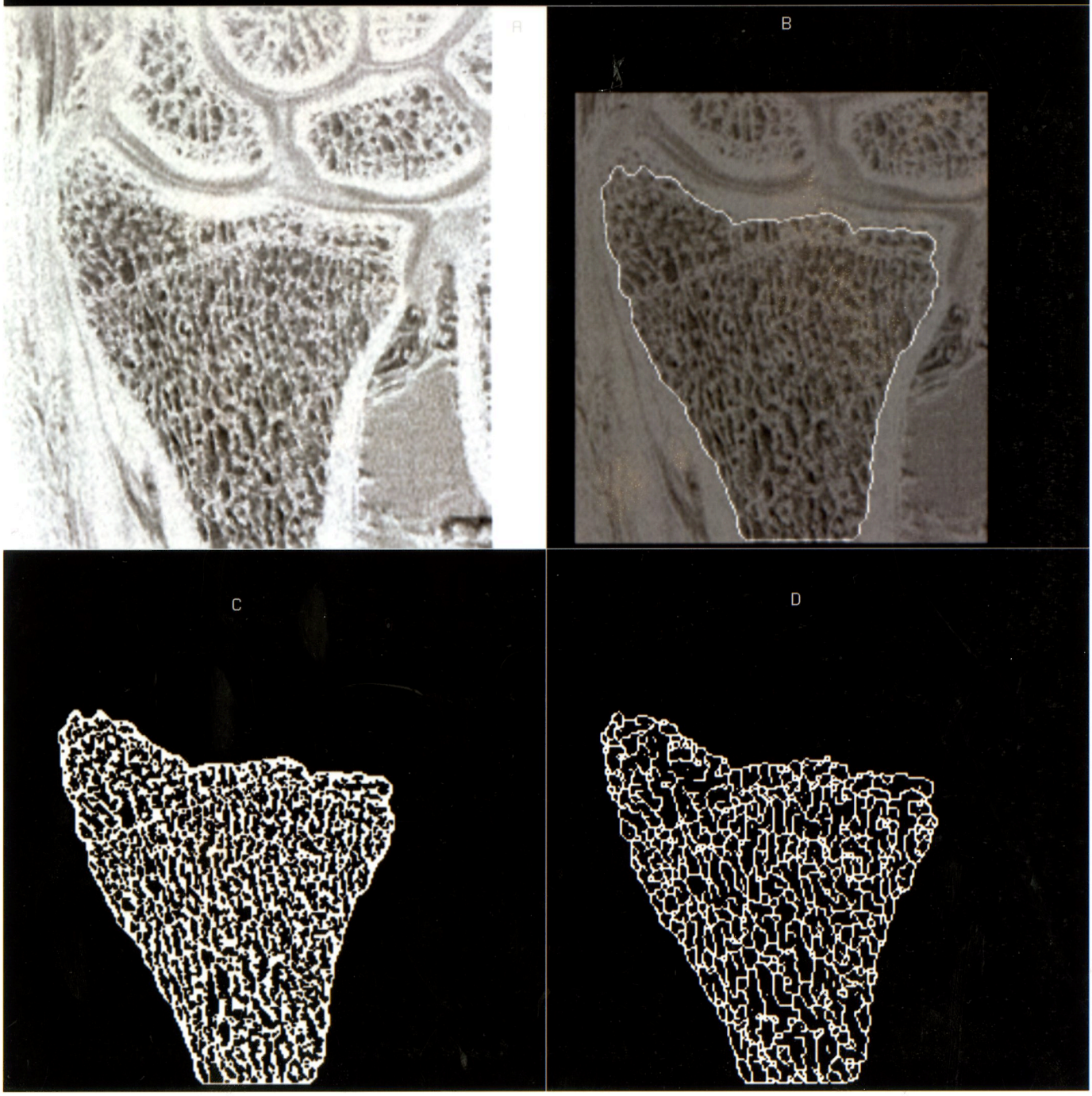
5.4 Indices of Structure

Connectivity was again assessed by the connectivity index (CI) defined in equation 5.1 and marrow space distributions were assessed in terms of mean, median, and maximum hole area. Unlike pQCT, MR allows for the examination of structure in each of the three orthogonal planes. A means of quantifying trabecular bone orientation is introduced to make use of this added information.

5.4.1 Marrow hole areas and connectivity

To obtain the indices of hole size, the postprocessing algorithm records the number and area of each hole present in the

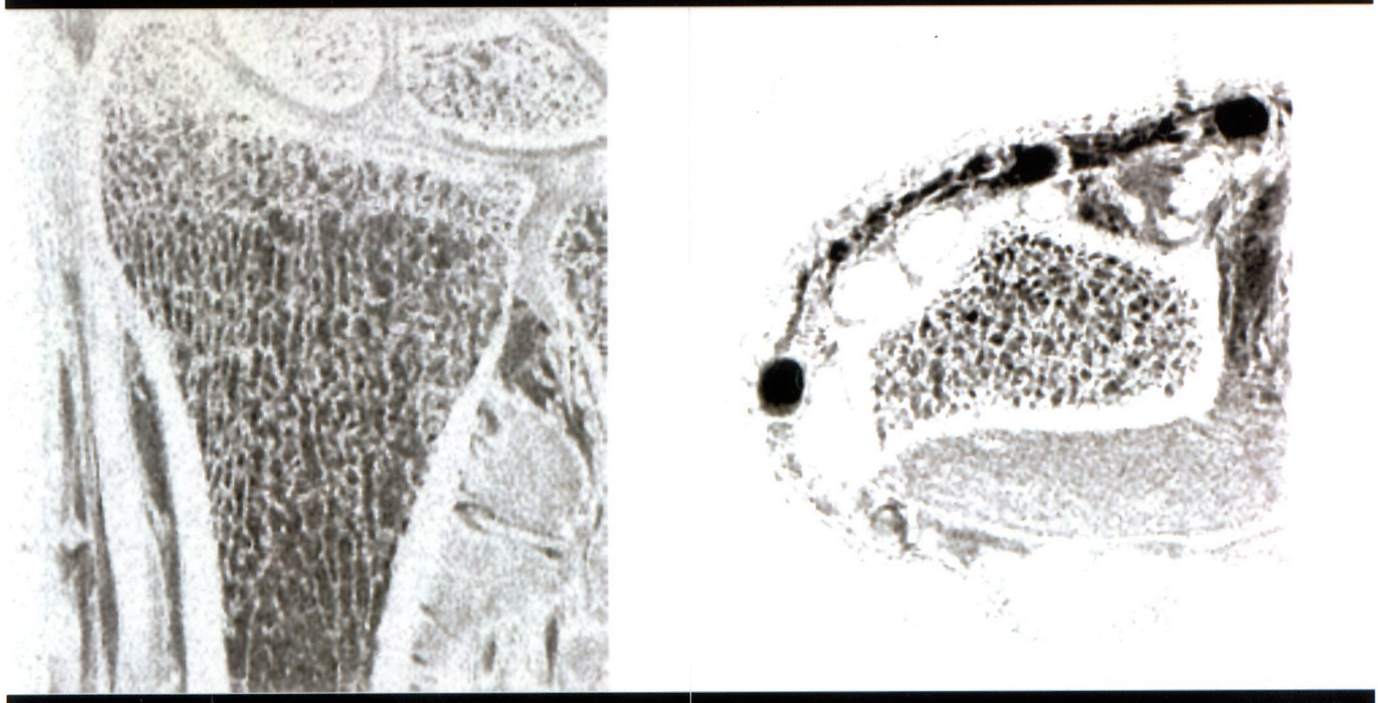
Figure 5.6: The postprocessing steps used to assess trabecular bone structure at the distal radius are shown. The original image (A) is of a healthy 29 year old male. The image is displayed as a negative image so trabeculae appear white while the normally high signal from fat appears dark. The bone structure is segmented by defining the boundry between cortical and trabecular bone (B). The trabecular network is reduced to a binary image (C) which is thinned to produce a representation from which connectivity can be assessed (D).



binary representation of the trabecular structure. Therefore, this allows for the distribution of hole sizes to be examined. To illustrate this, figure 5.7 shows the appearance of the trabecular structure at the distal end of the radius of a 48 year old male when imaged in the coronal(a) and axial planes(b). A total of 321 holes were detected in the coronal image. The mean hole area was 1.29 mm^2 while the median value was 0.19 mm^2. A total of 179 holes were detected in the axial image. The mean hole area was 0.51 mm^2 and the median value was 0.27 mm^2. The distribution of the hole sizes derived from these two images is plotted in figure 5.8. Two points are worth highlighting from this plot. First, the shape of the distribution is consistent with past histological findings. The areas of most holes are less than 0.5 mm^2 but there is a wide variation. Second, a mean hole size rather than a median value may best characterize the mechanical competence of the imaged bone. Although the majority of holes are only a few pixels in area the presence of a few large holes resulting from breaks in the network skews the average value. This estimate of the marrow space (H_A) is similar to the star volume parameter which has been derived from biopsy samples (Vesterby et al 1989a). However, in contrast to the star volume index, quantitation of the marrow space by H_A and H_M is derived in-vivo and requires little operator intervention.

Figure 5.9 shows the thinned binary representations of the trabecular bone network at the distal end of the radius of a 34 year old female and a 61 year old female. The thinned network of the older female clearly reveals areas of complete trabecular bone loss. As noted, the connectivity indices of the 34 year old

Figure 5.7: The orthogonal trabecular bone structure at the distal end of the radius of a healthy 48 year old male subject. The structure is viewed coronally (A) and cross-sectionally (B). The images are again displayed in inverse grey scale so that bone appears bright and fat dark.



A

B

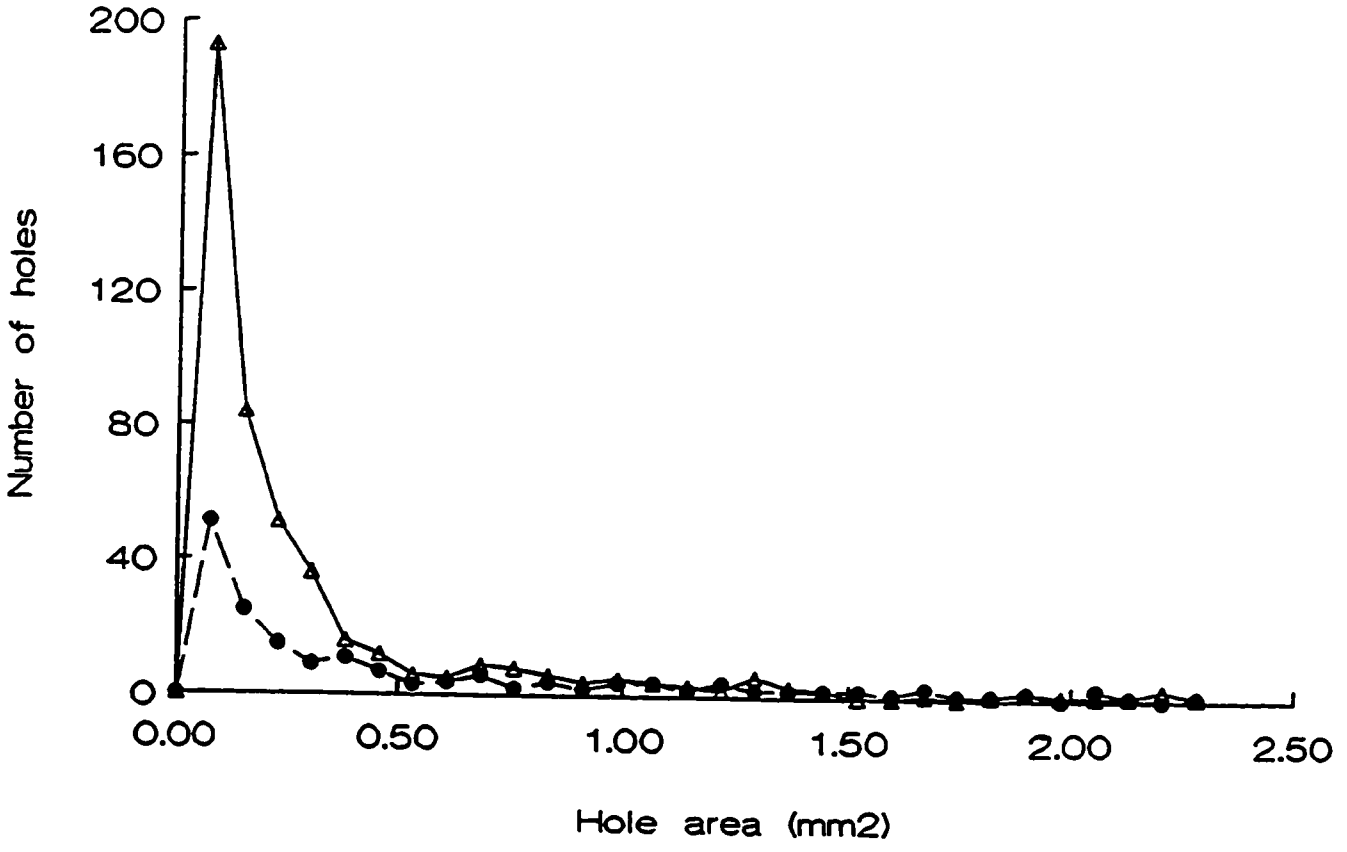


Figure 5.8: The distribution of hole sizes derived from the two images shown in figure 5.7. The solid line joining (▲) is derived from the coronal image while the dashed line joining (●) is derived from the cross-sectional image.

and 61 year old subject are 7.44 and 1.12, respectively. Differences in the integrity of the trabecular structure are also apparent when the mean hole area (H_A) is compared in these two subjects. For the 34 year old subject H_A is 1.15 mm^2. The value increases to 1.82 mm^2 for the 61 year old subject. To further illustrate the differences between these two subjects, figure 5.10 shows the distribution of nodes and free ends in the thinned binary representations which are displayed in figure 5.9. Since nodes and free ends are shown in yellow and red, respectively, the presence of more red points in the image of the older subject is a clear indication of a highly disrupted network.

5.4.2 Orientation

Trabecular orientation was assessed by evaluating the gradient at each point within the contoured bone area (figure 5.6b) using the scheme suggested by Caldwell (1995). This involved convolving the image with an edge detection mask. We chose to apply the Sobel mask (Gonzalez and Woods 1995). As the edge detection mask is convolved through the image the magnitude and direction of the gradient at each point in the image is calculated. This calculation involves the following steps. The point in the image at which the gradient is calculated can be denoted by F(i,j) and its eight nearest neighbours are numbered with the following scheme:

$$\begin{matrix} A_0 & A_1 & A_2 \\ A_7 & F(i,j) & A_3 \\ A_6 & A_5 & A_4 \end{matrix}$$

The gradient (G) at each point F(i,j) in the image is then given as:

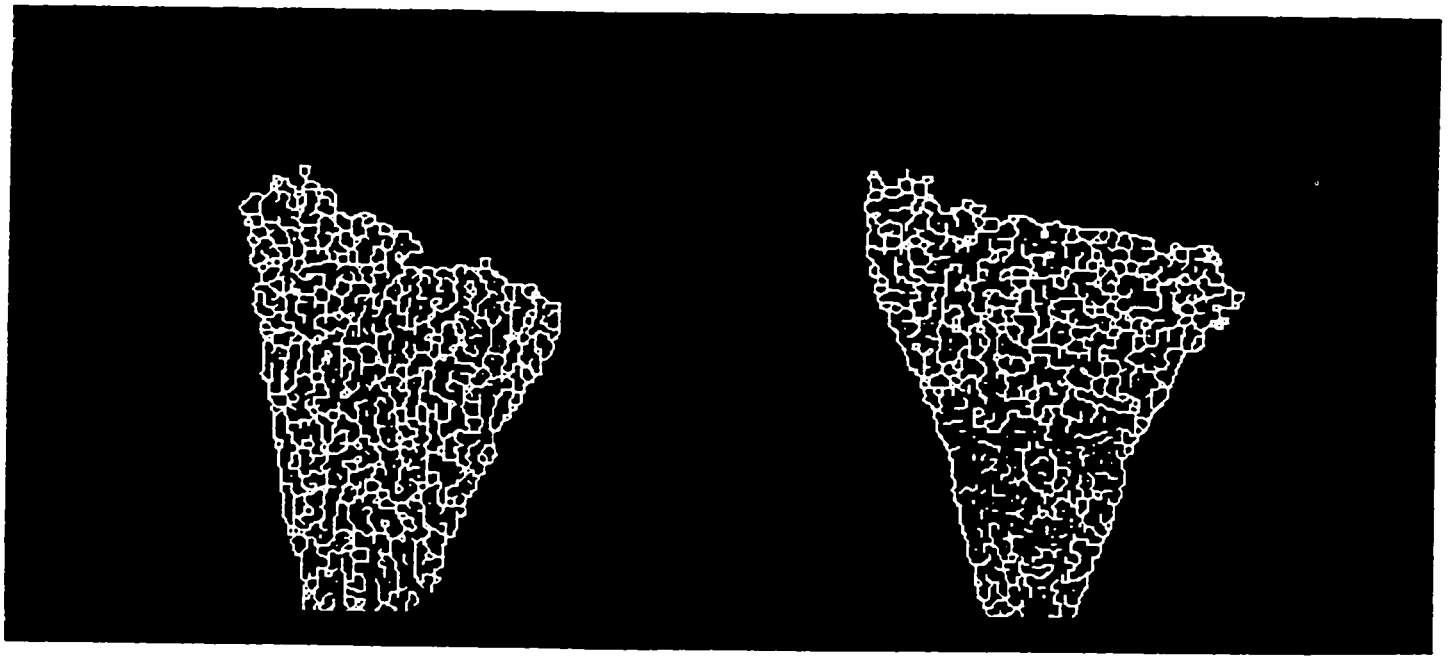


Figure 5.9: The skeleton representation of the trabecular bone network at the distal end of the radius of a 34 year old female (A) and a 61 year old female (B). A CI value of 7.44 was determined for the 34 year old and 1.12 for the 61 year old subject.

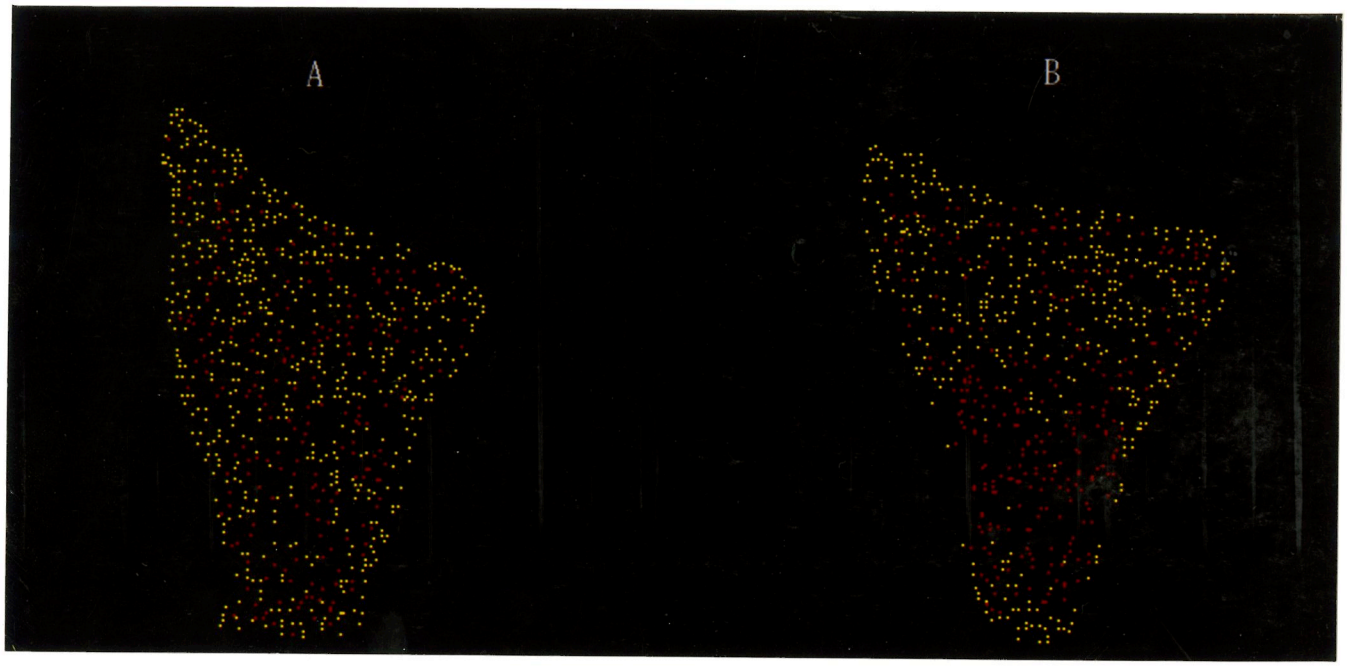


Figure 5.10: The distribution of nodes (yellow) and free ends (red) in the trabecular bone network at the distal end of the radius of a 34 year old female (A) and a 61 year old female (B).

$$G(i,j)=\sqrt{(x^2+y^2)} \qquad 5.2$$

where

$$x=(A_2+2A_3+A_4)-(A_0+2A_7+A_6)$$
$$y=(A_0+2A_1+A_3)-(A_6+2A_5+A_4)$$

The direction (Θ) associated with the gradient G at each point in the image is given by:

$$\theta=\tan^{-1}\left(\frac{x}{y}\right) \qquad 5.3$$

where Θ can range from 0° to 180°.

A trabecular bone network with trabeculae oriented at one of three angles (45°, 90°, 135°) is sketched in figure 5.11. In the orientation we have chosen to image the radius, structures with a gradient angle of 90° run parallel to the long axis of the bone while those at 0° or 180° run orthogonal to the long axis of the bone. Note that this co-ordinate system is also displayed with the bone network sketched in 5.11. If this structure were analyzed for orientation using gradient analysis then the analysis would proceed through the following steps which are also indicated in figure 5.11. G and Θ are calculated at each pixel in the image. If the magnitude of the gradient at a given pixel exceeds a defined threshold, the appropriate bin is incremented by unity. The angular bins are normally 5° wide. The histogram that results displays the frequency with which the trabecular elements are oriented along a given direction. The gradient frequencies are normalized to the sum of all frequencies detected over all directions to express orientation as a percentage. If the gradient frequency at a given angle Θ is denoted H_Θ then mathematically this normalization can be expressed as:

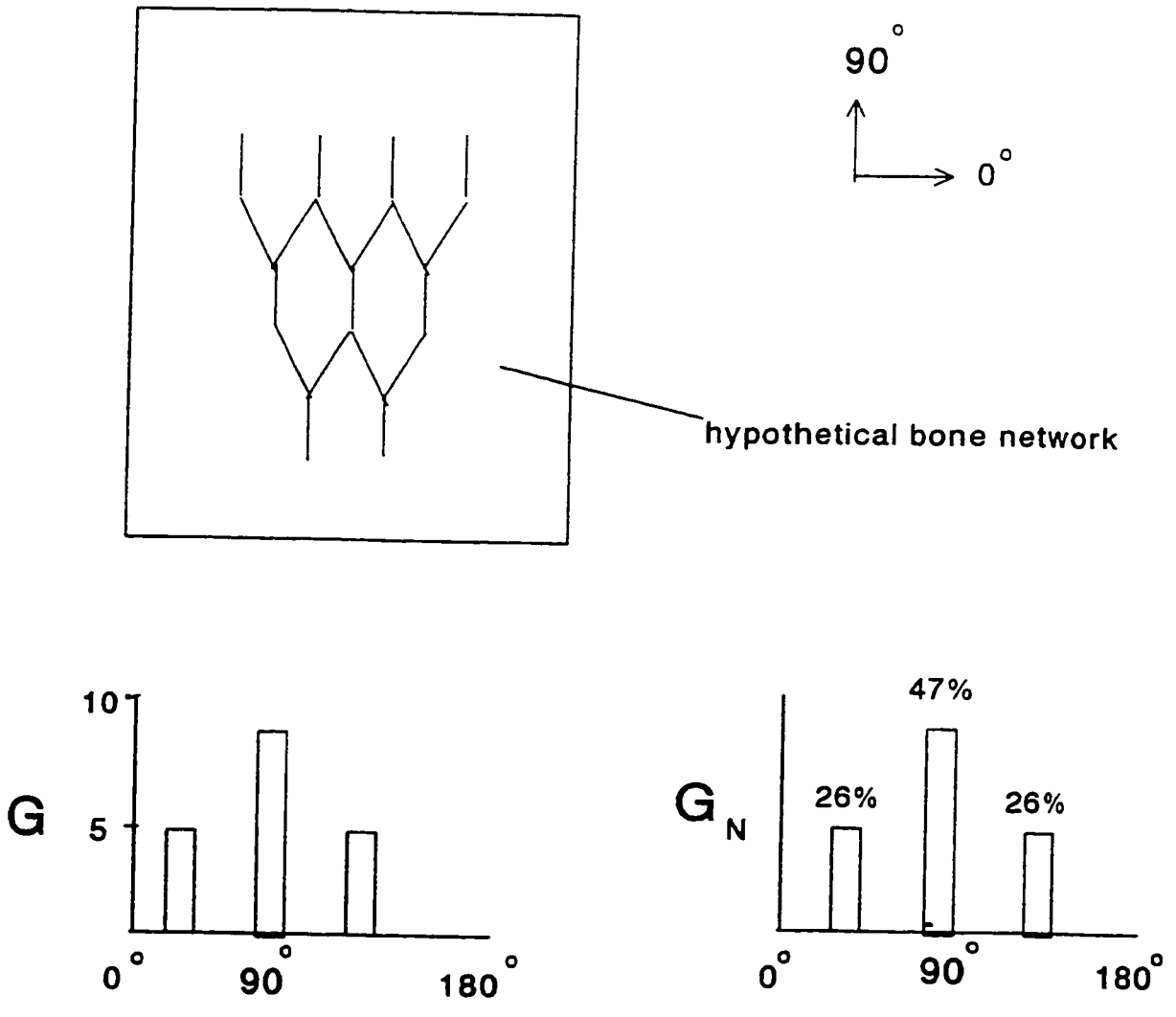


Figure 5.11: An illustration of the application of gradient analysis to a theoretical bone network. Note that 9 trabeculae are oriented at 90°, 5 at 45° and 5 at 135°. This corresponds to peaks of magnitude 47%, 26%, and 26% respectively in the normalized gradient histogram (G_N).

$$G_N=\left(\frac{H_\Theta}{\sum H_\Theta}\right)\cdot 100 \qquad 5.4$$

where the normalized gradient frequency is given the acronym G_N and the summation is performed from 0° to 180°.

To allow comparison among subjects a new parameter G_I is introduced. It is defined from the distribution of G_N. G_I is defined as the area under the curve given by G_N integrated from 75° to 105°. Mathematically this integration can be expressed as:

$$G_N=\sum G_I(\theta) \qquad 5.5$$

where the summation is performed from 75° to 105°. To illustrate the use of this new index of orientation (G_I), the result of applying gradient analysis to the images in figure 5.7 is plotted in figure 5.12. Three points are worth highlighting from these plots. First, the area under each curve is 100%. Second, the magnitude and location of a peak in the histogram plot indicates the degree of anisotropy present in the trabecular structure imaged. Therefore, the presence of a peak at 90° in the plot derived from the coronal image indicates that individual trabeculae are preferentially oriented along the long axis of the radius. The absence of a peak in the gradient histogram derived from the axial image indicates that there is no preferential orientation of the trabeculae that run orthogonal to the long axis of the bone. Third, this structural preference is reflected in the G_I values derived from each curve. For the coronal image G_I is 42%. It decreases to 19% for the cross-sectional image. This structural pattern is consistent with the nature of the forces acting on the wrist. In general, when the wrist is

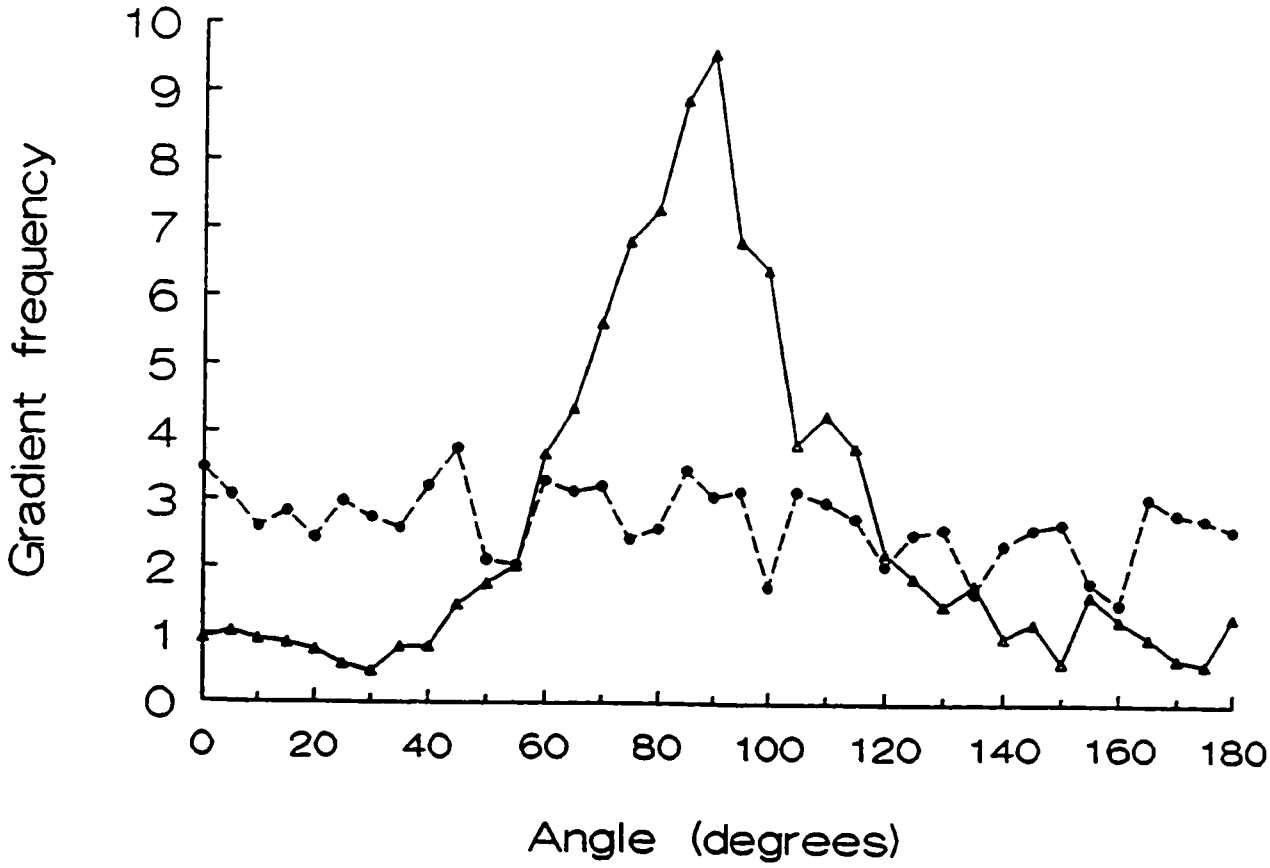


Figure 5.12: The frequency of occurrence of gradient magnitudes as a function of angle for the two images shown in figure 5.7. The solid line joining (▲) is derived from the coronal image while the dashed line joining (●) is derived from the cross-sectional image.

subjected to a mechanical load the force will act along the radius rather than across it. Therefore it is not surprising that any architectural anisotropy disappears when the bone is viewed on cross-section. This response of trabecular bone to applied forces can be summarized by Wolff's law which states that the orientation, structure, and strength of bone at a given anatomic location is a direct response to the mechanical stresses resulting from normal joint function.

Others have illustrated the value of analyzing trabecular orientation by gradient analysis. For example, using digitized plane film X-rays, Rockoff (1971) showed that at the distal end of the radius the orientation of the trabeculae in normal bone is more longitudinal than that in diseased bone. More recently Caldwell (1995) showed that an index of trabecular orientation derived from gradient analysis distinguishes the compressive strength of strong from weak vertebrae in-vitro better than bone density.

5.5 Summary

The structural parameters derived by the algorithm are not strictly dependent on trabecular size but on the connectivity of the bone structure. This is appropriate since bone mass is lost by removal of entire trabeculae rather than by a generalized uniform thinning of the whole trabecular bone network. The resulting space left by lost trabeculae is filled by fatty marrow and remaining trabeculae are more widely separated, less connected, and therefore less likely to withstand a compressive force. Such patterns of bone loss may best be highlighted by

examining the size of the marrow spaces and the connectivity of the bone network. The postprocessing algorithm is able to record indices of trabecular spacing (H_A) and trabecular connectivity (CI). These two indices may change in a manner consistent with bone loss through aging and be sensitive to differences in the mechanical integrity of trabecular bone. This sensitivity is explored in the remaining chapters.

Chapter 6

IN-VIVO ASSESSMENT OF TRABECULAR BONE STRUCTURE FROM pQCT IMAGES

6.0 Introduction

This chapter presents and discusses the results of applying the indices of structure proposed in chapter 5 to a group of normal volunteers with and without low bone mass and a small group of patients who have suffered a Colles (wrist) fracture. The discussion begins with a description of the two study groups. This description is followed by an examination of the intra-subject variability associated with an assessment of structure, and is followed by an evaluation of the structural variations along the radius. The chapter ends by comparing the specificity and sensitivity with which the Colles fracture patients can be discriminated from normal on the basis of density and structure.

6.1 Results of Pilot Study

A total of 31 female and 13 male subjects underwent a pQCT evaluation in this study. They were referred for a pQCT scan from the Fracture Clinic operated at St Joseph's Hospital in Hamilton. They were classified as either normal (23 subjects) or having suffered a fracture of the radius (21 subjects) as determined by a plane film x-ray. The normal subjects ranged in age from 21 to 77 years. Their trabecular bone density (TBD) at the radius ranged from 136.6 to 336.8 mg cm^{-3}. Their cortical bone density (CBD)

ranged from 481.8 mg cm^{-3} to 952 mg cm^{-3}. The fracture subjects were aged 40 to 82 with a TBD range of 86.8 to 235.2 mg cm^{-3} and a CBD range of 449.1 mg cm^{-3} to 970.5 mg cm^{-3}. Trabecular bone density, cortical bone density, and structure was assessed in the non-fractured arm in those patients identified with a fractured wrist.

6.1.1 Reproducibility

For all practical purposes, the value of the indices derived from a given image were not dependent on the size of the rectangular ROI placed around the radius by an operator. For example, as examined by defining ROI's of various size for a given image, a 20% change in ROI area caused less than a 1% change in both CI and H_A.

A second issue of reproducibility concerned repeat measurements with repositioning. This intra-subject variability was assessed by performing three scans on two subjects over a 3-month period. Mean coefficients of variation of 5.1% and 5.5% were found for CI and H_A. These levels of reproducibility for CI and H_A are higher than that reported for the determination of bone mineral density in the appendicular skeleton (~1%) (Muller et al 1989). Instead of a direct comparison with bone density it would be more useful to compare these levels of intrasubject variability in CI and H_A with similar indices of structure assessed from biopsy. Such a comparison, however, is not possible because a second sample cannot be taken from a site already biopsied.

6.1.2 Structural variations along the radius

Differences in trabecular architecture could be identified from the cross-sectional slices acquired at different points along the radius. The dependence of connectivity and marrow pore size upon the location of the pQCT image slice along the radius is shown in table 6.1. These results were recorded for a young male subject and indicate the importance of positioning. As the image location proceeds proximal from the head of the radius, the connectivity index decreases while the mean hole area and the maximum hole area increase. For the image slice recorded at 70 mm from the head of the radius the mean hole area corresponds to the area enclosed by the inner contour of the cortical bone shell. The negative CI value recorded at this slice position is an indication of the irregularity of the inner surface of the cortex. These irregularities project into the marrow cavity, thereby being recorded as free ends. This data is consistent with past histological studies. It has been demonstrated that proceeding proximal from the distal end of the radius, the fraction of trabecular bone increases over approximately the first 2 centimetres and then decreases rapidly to zero over the next 5 centimetres (Schlenker and VonSeggen 1976). This pattern is due to a corresponding change in the total number of trabeculae present which results in a decrease followed by an increase in the size of the intertrabecular spaces. This is exactly the trend identified by H_A.

Table 6.1. The variations in connectivity (CI) and marrow hole area along the radius of a normal male volunteer.

Distance from head of radius (mm)	Connectivity index (CI)	Mean hole area (H_A) (mm^2)
12.4	29.67	0.53
17.4	26.69	0.57
22.4	25.22	0.69
27.4	22.56	0.73
70.0	-6.59	18.95

6.1.3 Correlations

Pearson correlations between the indexes of bone structure and trabecular density and age are listed in table 6.2. The indexes were also highly intercorrelated. For example, CI showed a strong ($-0.8 < r < -0.9$) but negative correlation with H_A and H_M. The network length (NL) was strongly correlated with both the total bone area ($r=0.77, p=0.0003$) and trabecular bone area ($r=0.64, p=0.006$). This confirms that in our original definition of CI, division by NL accounts for different bone sizes. These positive and negative correlations between trabecular density and our indices of structure reflect known patterns of bone loss established from biopsy studies. For example, in the normal aging process of the adult skeleton, bone mass is lost from trabecular bone. This is due to entire trabeculae being removed rather than a generalized uniform thinning of the whole trabecular structure. The resulting space left by the lost trabeculae is filled with fatty marrow and those trabeculae that remain are more widely separated, less connected, and therefore less likely to withstand a compressive force (Parfitt et al 1983). Consequently, indices related to the marrow space change inversely with density while those relating trabecular connectivity will vary directly with density.

6.1.4 Discrimination of the two groups

The range of trabecular bone density and cortical bone density in the non-fractured and fractured groups are indicated in figure

Table 6.2. Correlations between indices of structure and bone density and age. Correlations with age were not significant while those with density were highly significant ($p<0.001$).

	CI	H_A	H_M
Age	-0.315	0.187	0.182
Trabecular density.	0.855	-0.735	-0.831

6.1 and figure 6.2, respectively. Figure 6.2 shows that cortical bone density has no merit in distinguishing fractured patients from non-fractured subjects. For this small study population the threshold of low bone density is defined as the mean TBD value minus 2 SD in our 23 non-fractured subjects. The mean (220 mg cm^{-3}) and low bone density threshold (138 mg cm^{-3}) are also indicated in figure 6.1. This trabecular density threshold separates the fractured from the non-fractured subjects with a sensitivity of 38% and a specificity of 100%. Figure 6.3 shows that the difference between the two groups is enhanced by the connectivity index. Figure 6.4 shows the same for the mean hole area. A CI threshold (mean CI minus 2 SD) of -7.6 achieves a sensitivity of 48% and a specificity of 96%. A threshold (mean plus 2SD) of 4.4 mm^2 for H_A achieves a sensitivity of 67% and a specificity of 96%. These findings suggest that there is diagnostic value in the proposed indices in predicting radial fractures in a small group of subjects. Both CI and H_A accurately identify subjects with wrist fractures in this mixed study population with a greater sensitivity than trabecular bone mineral density and cortical bone density. The poorer performance of bone density in identifying those who have fractured verifies that bone failure is a complex disorder that cannot be predicted by measuring bone mass alone. Moreover, our results emphasize that assessing bone architecture can clarify a significant portion of this complexity.

Figure 6.5 shows the thinned binary representation of the bone structure at the distal radius typical of a normal middle aged

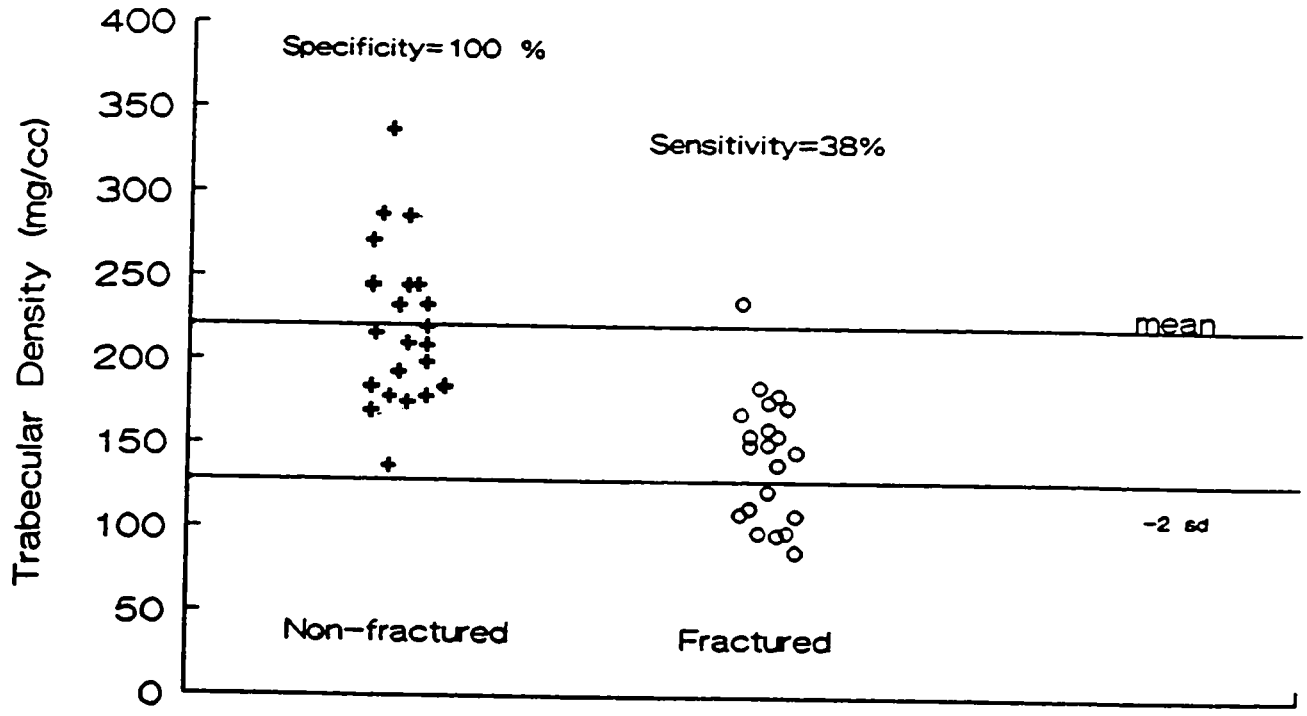


Figure 6.1: Comparison of the range of trabecular bone densities recorded in the non-fractured (+) and fractured (o) groups. The mean density and 2 sd below the mean density in the non-fractured group is indicated.

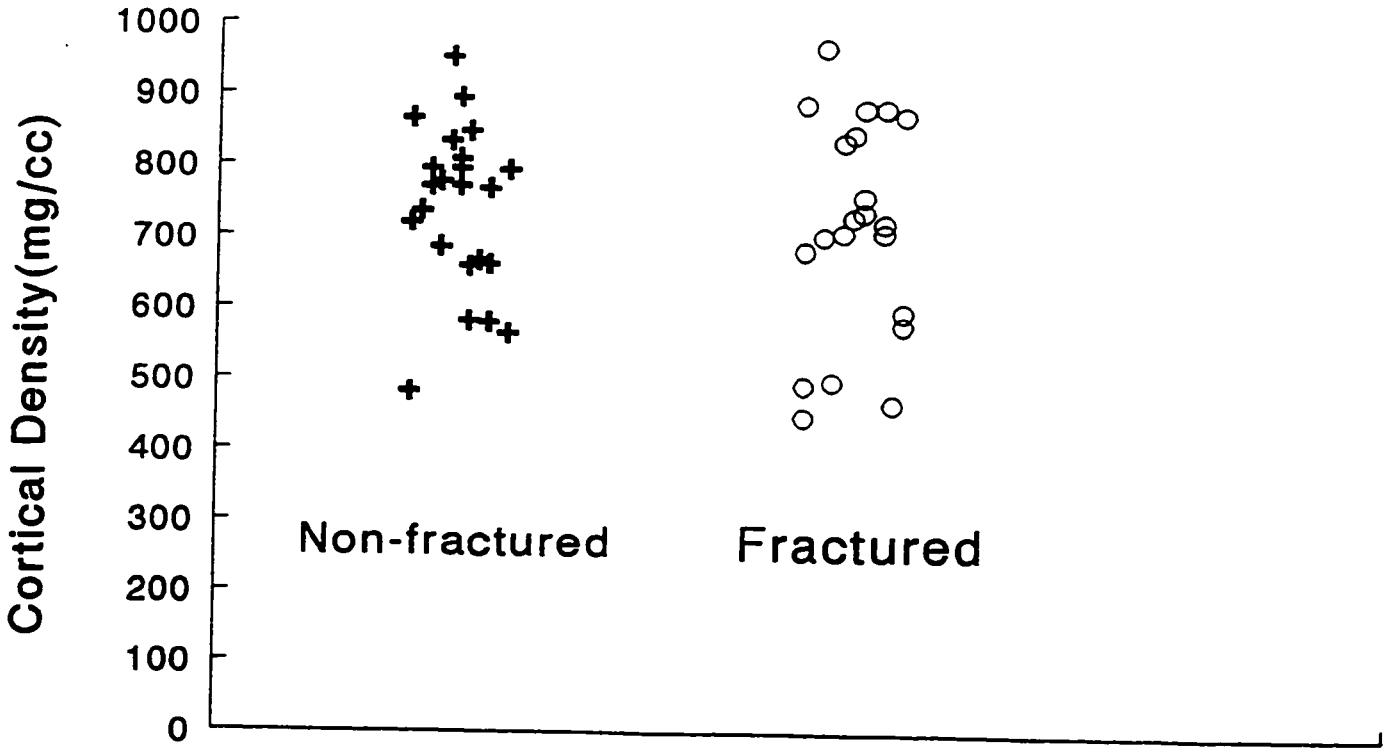


Figure 6.2: Comparison of the range of cortical bone densities recorded in the non-fractured (+) and fractured (o) groups.

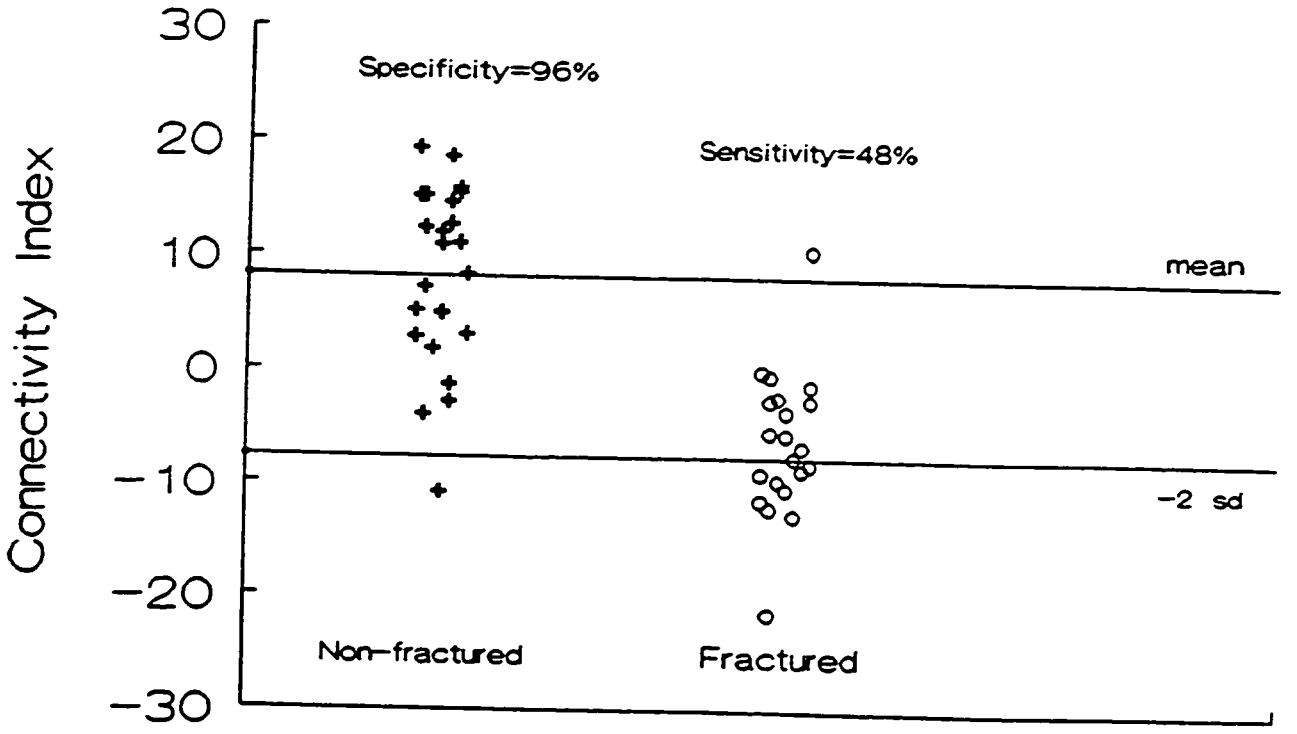


Figure 6.3: The division of non-fractured (+) and fractured (o) subjects based on a measurement connectivity (CI). The mean CI and a threshold at 2 sd below the mean in the non-fractured group are indicated.

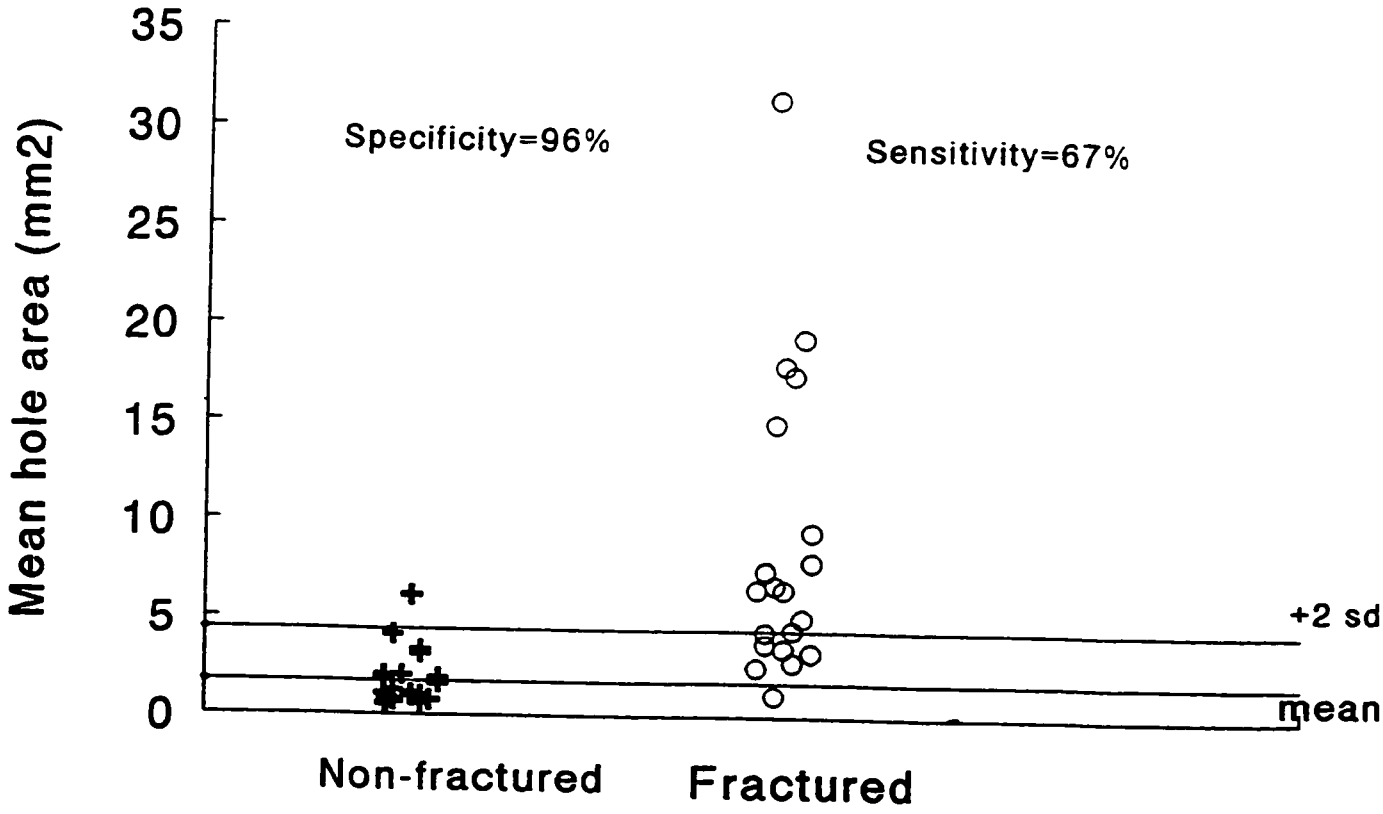


Figure 6.4: The division of non-fractured (+) and fractured (o) subjects based on a measurement of mean hole area (H_A). The mean H_A and a threshold at 2 sd above the mean in the non-fractured group are indicated.

female and an elderly female who suffered a wrist fracture. Keeping in mind that the non-fractured wrist was measured in the elderly subject, for a 15% difference in trabecular bone density there is a 36 fold difference in the connectivity index. Large differences are also apparent when the mean hole area (H_A) is compared in these two subjects. For the middle aged subject H_A is 1.07 mm^2. The value increased to 4.17 mm^2 in the elderly subject.

6.2 Summary

A dedicated computed tomography system was used to acquire transaxial images of the distal radius to assess trabecular bone structure in-vivo. Trabecular bone was segmented from the marrow and soft tissue background by postprocessing the image with a region grow and skeletonization step. From the processed image the integrity of the bone was assessed by examining the continuity of its trabecular network and by determining the area of the holes comprising its marrow space. The continuity of the bone imaged was assessed by a proposed connectivity index (CI) and the size of the marrow spaces was assessed by calculating a mean hole area (H_A) in the bone cross-section. Repeat measurements revealed that the intra-subject variability in CI and H_A was small (CV<6%). Both CI and H_A were sensitive enough to reflect differences in structure at the head of the radius and at several sites along its shaft.

The diagnostic value of assessing bone structure at the distal end of the radius was tested by measuring trabecular bone density, cortical bone density, CI and H_A in a mixed group of 44 subjects,

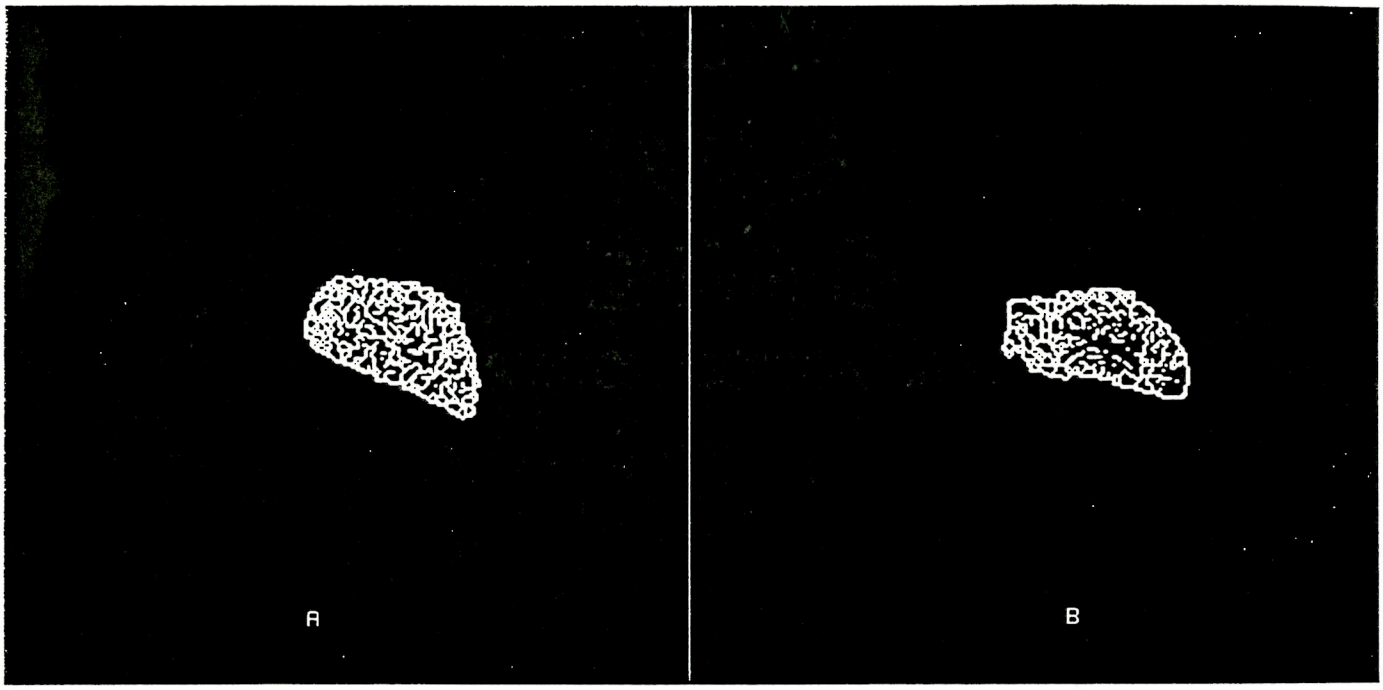


Figure 6.5 This figure shows the thinned binary representation of the connectivity in the trabecular bone network at the distal radius of a normal 48 year old female with a trabecular density of 183.8 mg cm^{-3} (**A**) and a 69 year old female with trabecular density 155.7 mg cm^{-3} who experienced a wrist fracture(**B**). A CI values of 14.7 was determined for the 48 year old and 0.41 for the 69 year old.

21 of whom had suffered a wrist fracture. A measure of cortical bone density revealed no difference between the two groups. However, a trabecular bone density threshold of 138 mg cm^{-3}, corresponding to 2 standard deviations below the mean density in the 23 non-fractured subjects separated fractured from non-fractured subjects with a sensitivity of 38% and a specificity of 100%. A CI threshold of -7.6 increased the sensitivity (48%) and almost maintained the 100% specificity. An H_A threshold of 4.4 mm^2 achieved a sensitivity of 67% and a specificity of 96%. This increased sensitivity achieved by the indices of structure suggests that an in-vivo assessment of trabecular bone structure can contribute significantly to the identification of persons at risk of fracture.

Chapter 7

IN-VIVO ASSESSMENT OF TRABECULAR BONE STRUCTURE FROM HIGH-RESOLUTION MAGNETIC RESONANCE IMAGES

7.0 Introduction

To be useful clinically, structure must be imaged in-vivo with thin slices (< 1 mm), multiple slices, small pixel sizes (<0.5mm), and conveniently short scan duration (< 20 minutes). Two factors inhibit accurate evaluations of standard histomorphometric parameters of trabecular bone structure from MR images. First, the minimum slice thickness allowed by the gradient fields on current whole body imagers is 2 to 3 times thicker than the average trabecular width. Therefore, projections blur the bone structure in the final image. Thinner slices can be achieved in high-field small bore systems. For example at 9.4 T, slice thicknesses of 100 μm are readily achieved. However, sample volumes are limited to 1 cm^3, imaging times of 1-2 hours are required, and only biopsy samples can be imaged (Wehrli et al 1991). Second, at the interface between trabecular bone and marrow, non-linear magnetic field gradients are established which reduce the apparent spin-spin (T_2) relaxation rate of fat protons near the bone. The T_2 reduction makes these protons invisible and results in an increase in the apparent width of the bone (Sebag and Moore 1990, Wehrli et al 1991, Jara et al 1993). This effect is depicted in figure 7.1. An error in trabecular width may be present in both spin echo and gradient echo images. Even with these imaging constraints, high-resolution MR, coupled with

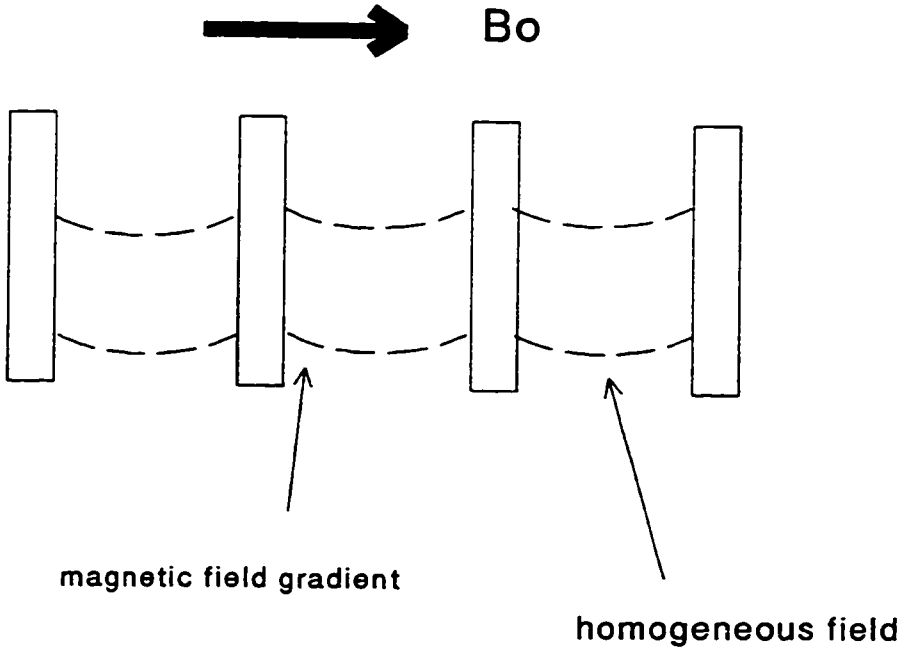


Figure 7.1: Effect of magnetic susceptibility on the appearance of individual trabeculae in gradient echo images. The presence of magnetic field gradients which are established at the boundary between bone and marrow reduces the signal from marrow protons. The result is an increase in the apparent width of trabecular bone.

image processing methods can be used to assess bone structure at peripheral sites such as the wrist. However, poor SNR must be overcome before the trabecular architecture can be revealed by MR.

This chapter, first of all, examines the scanning parameters which can be manipulated to improve image quality by maximizing the signal to noise ratio and minimizing the likelihood of blurring due to subject motion. Once acquired, the high-resolution MR images are segmented using the post-processing steps described in chapter 5. The chapter ends by deriving indices of trabecular connectivity and orientation at the distal end of the radius for a mixed population of normal volunteers.

7.1 Choice of Scan Parameters

All images were acquired on a 1.5 Tesla General Electric Signa clinical imager running version 5.43 software. A three dimensional (3D) gradient echo scan prescription is the most suitable pulse sequence to image the structure at the wrist. The reasons for this are as follows. Three dimensional pulse sequences excite a slab of tissue and the slab can be divided into a series of contiguous slices by appropriately switching the three gradient fields. The manner in which the data is acquired allows for the tissue volume to be divided into very thin slices (< 1mm) which cannot be achieved with two dimensional acquisitions. The GE Signa software can divide a 3D volume into 12 to 60 contiguous slices each with a minimum slice thickness of 700 μm. These very thin slices are essential for in vivo structure assessments. Both gradient echo and

spin echo sequences can be used to image trabecular bone. However, 3D spin echo scan prescriptions are not available as a pulse sequence option because the deposition of radio frequency power limits the minimum slice thicknesses which can be achieved in vivo. This limitation does not arise for gradient echo sequences because spin rephasing is achieved by switching magnetic field gradients rather than by the delivery of a 180° rephasing radio frequency pulse. Hence, for in vivo clinical applications, a 3D gradient echo pulse sequence must be used to obtain high resolution images of trabecular bone structure at peripheral sites.

Without the aid of specialized local gradient coils or modified pulse sequences, a number of factors can be adjusted to improve the visualization of trabecular bone structure at the wrist. For example, the SNR depends on many factors of operation such as type of receive coil, echo times, matrix size, slice thickness, field of view, number of signal averages, and receiver band width (Wood et al, 1993). The manner in which some of these factors relate to SNR can be described by equation 7.1:

$$SNR = k\frac{FOV_{freq} \cdot FOV_{phase} \cdot SL \cdot \sqrt{Nex}}{\sqrt{N_{freq} \cdot N_{phase} \cdot BW}} \qquad 7.1$$

where FOV_{freq} is the field of view in the frequency encoding direction, FOV_{phase} the field of view in the phase encoding direction, SL the slice thickness, Nex the number of signals averaged, N_{freq} the number of pixels across the FOV_{freq}, N_{phase} the number of phase encoding

steps, and BW the receive band width. The constant k is included as a scaling factor. As shown in equation 7.1, one factor which can be changed to improve SNR is the receive band width. The receive bandwidth refers to the range of frequencies within which the system will respond. The Signa software offers selections from 2 kHz to 32 kHz. The system default is ±16 kHz for a 256 frequency matrix and ±32 kHz for a 512 frequency matrix. This means, for example, that the system will detect signal from protons resonating at frequencies anywhere in the range of ±16 kHz from the centre frequency for a 256 frequency matrix. When the band width is narrowed, the system looks at a smaller range of frequencies. This has the benefit of excluding much of the random electronic noise inherent in an MR system. The result is improved SNR. For example if the BW is halved SNR improves by a factor of 1.4. Because frequency differences are used to encode position, it is not surprising that narrowing the BW also restricts the minimum field of view (FOV). This link between BW and minimum FOV for the GE Signa system is shown in table 7.1. Although selecting the minimum BW (±2 kHz) would result in the greatest SNR improvement, the tradeoff is a substantial increase in imaging time. This is indicated by the minimum TR's allowed by the system. A 5 cm FOV was selected to reveal the bone structure at the distal end of the radius. Therefore, as indicated in table 7.1, a minimum BW of 9.20 kHz is linked with this FOV. To minimize scanning time a 256 square matrix rather than a 512 matrix was prescribed over the 5 cm FOV. Along with a factor of 2 improvement in SNR, this produces an in-

Table 7.1: The minimum values of BW, FOV and TR which are allowed by the GE Signa Advantage software when a 3D gradient echo scan sequence is prescribed. The values listed are for a 256 frequency matrix.

Minimum BW (± kHz)	Minimum FOV (cms)	Minimum TR (msec)
2.0	4	54
4.0	4	40
6.4	4	26
7.2	4	26
9.2	5	26
10.7	6	26
12.8	7	26
16.0	8	26

plane resolution of 197 μm which is sufficient to resolve individual trabeculae.

To overcome the poor SNR resulting from imaging at a 5 cm field of view a 3 inch receive only surface coil was used to detect the MR signal. Surface coils produce much higher SNR in small field thin slice studies because the coil can be placed close to the anatomy of interest. Also, depending on the size of the coil, it may reduce noise from outside the FOV. The work of Schenck (1984) illustrated the degree of SNR improvement with surface coils in comparison to body and head coils. A SNR comparison taken from this work is given in figure 7.2. It is clear from this plot that, up to a depth of 6 cms, surface coils produce better SNR than the others. Consequently, a 3 inch surface coil taped to the wrist was deemed the most appropriate for an examination of the bone structure at the wrist.

Ultimately it is the total imaging time available that controls the degree to which noise effects can be limited. For example, one of the simplest ways of reducing the level of noise in an image is to increase the number of signals averaged (Nex) to produce an image. Equation 7.1 reveals that, doubling the number of signals averaged increases the SNR by a factor of 1.4 but doubles the total imaging time. Shorter scanning times have the benefit of minimizing the likelihood of blurring due to subject motion. To reduce the likelihood of blurring, the high-resolution image set must be acquired in under 15 minutes. To offset these constraints of imaging time and SNR, the GE scanning software offers some

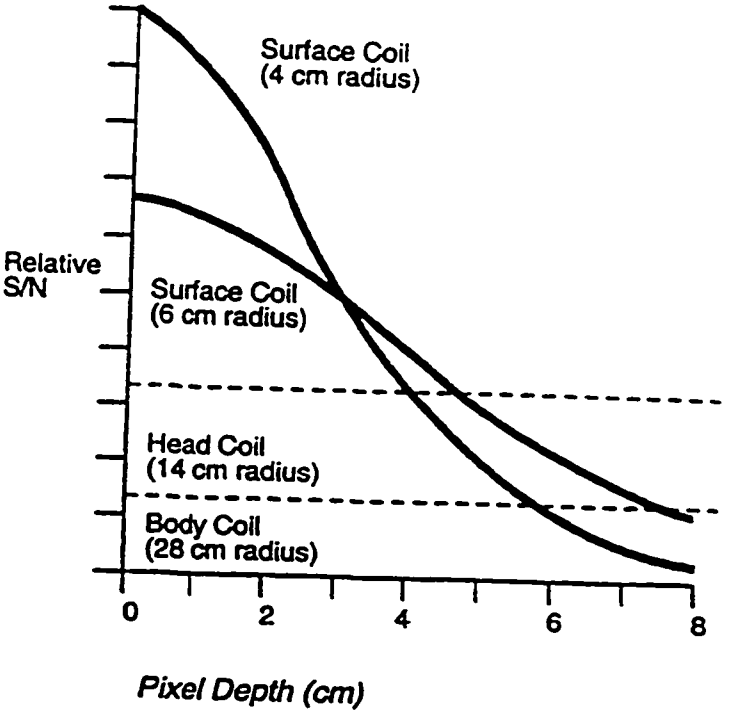


Figure 7.2: Comparison of surface coil SNR to that of body and head coils. Note that up to a 5 cm depth, surface coils will provide the best SNR (Schenck et al 1984).

flexibility in the choice of TE, TR, and Nex used to acquire the 3D data set. To aid in selecting the appropriate values a phantom was used. The phantom consisted of a 4 centimetre diameter cylindrical polyethylene container filled with corn oil to simulate fatty marrow. Three scan prescriptions were tested for their ability to maintain a high SNR if applied to in-vivo studies of trabecular structure at the wrist. Common to each of the three was that each was acquired with a fast gradient echo pulse sequence, a 60° flip angle, and an in-plane resolution of 195 μm x 195 μm x 800 μm (5 cm field of view). Each of the three scan sequences were obtained for 2 to 5 signal averages. The differences between the three sequences were as follows. The first was acquired at a BW of 9.20 kHz and a full echo was sampled during readout. A minimum TE of 17.5 msec was selected. For the second scan prescription the BW was kept at 9.20 kHz but a partial echo was sampled during the readout phase. The minimum TE allowed by the software was selected and was 11.8 msec. To sample a partial echo the system collects just part of the data that would normally be acquired to build an image. Interpolation schemes are used to fill in the missing data not sampled. With this approach, the TE is reduced but the effect of background noise is increased. For the third scan prescription a partial echo was sampled during readout but with a receiver BW of 4 kHz. The minimum TE for this prescription was 13.6 msec. The appropriateness of each of these three scan prescriptions for in-vivo studies was assessed by comparing the SNR derived from a common region of interest defined in images of the phantom. The mean and standard deviation

in the signal strength was calculated from the region of interest. The SNR was then expressed as the mean divided by its associated standard deviation. This dependence of SNR upon image acquisition time and TE for the gradient echo sequence used is plotted in figure 7.3. The increase in acquisition time corresponds to an increase in the number of signal averages prescribed during the pulse sequence. The SNR level for two through five signal averages is plotted on each curve. Images acquired at a TE of 17.5 msec showed a 15-20% improvement in SNR compared to those acquired at shorter echo times. However, the time required to acquire these images at a given number of excitations is doubled. Lowering the receiver bandwidth by a factor of two recovers some of the signal loss which results from sampling a partial echo but also increases the time required to obtain an image set by 25%. From figure 7.3, it is clear that the best SNR achieved in under 15 minutes was for a TE of 17.5 msec and at 3 Nex (a total imaging time of 12.5 minutes). These values were then used for the 3D gradient echo scanning protocol. This protocol is summarized in the following section.

7.1.1 Protocol for subject scans

The distal end of the radius was scanned in all subjects using a standard three inch circular surface coil. Subjects were placed in a supine position with their hands at their side and a series of axial and coronal spin echo sequences were used to locate the distal end of the radius. Once localized, a set of high resolution

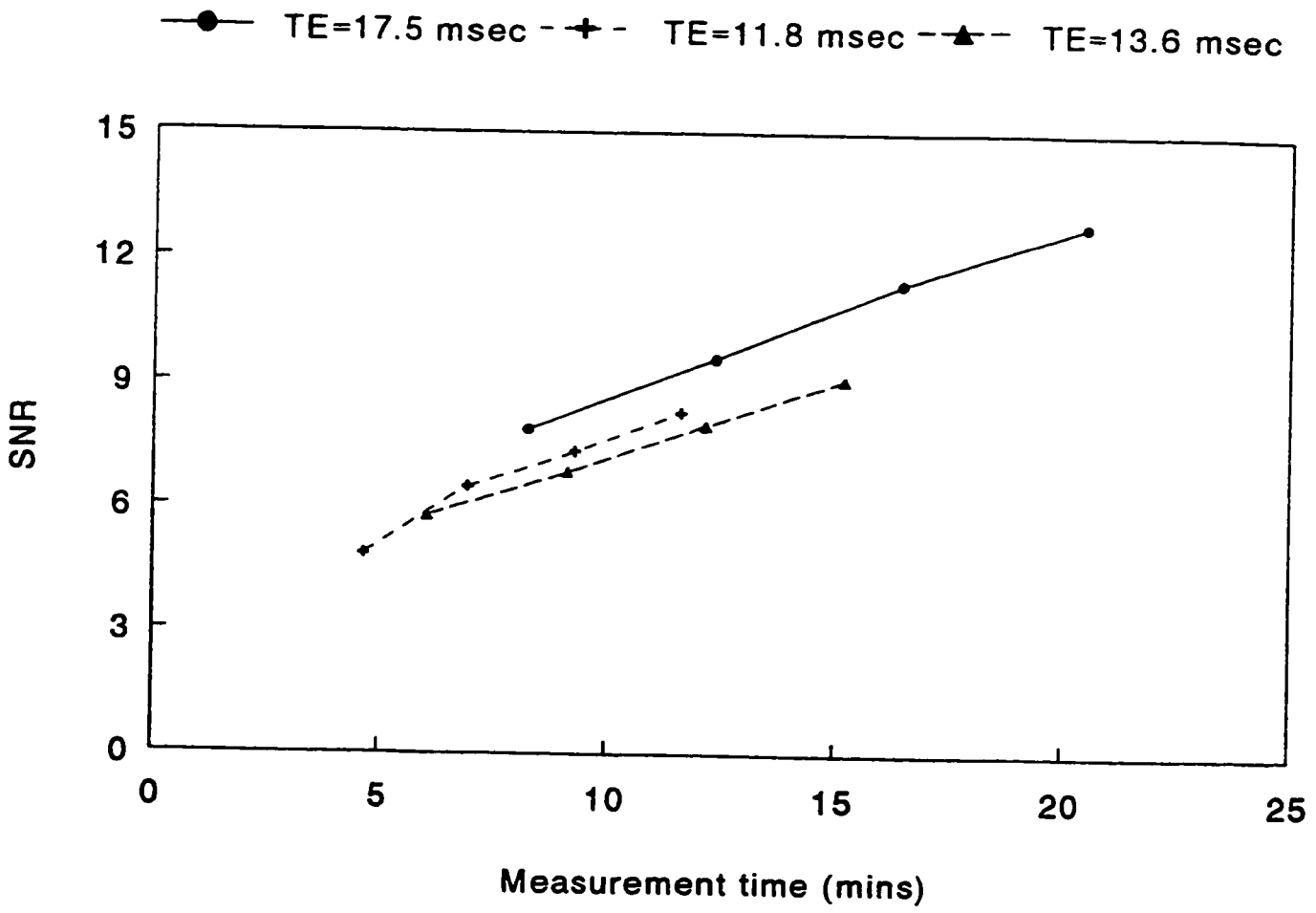


Figure 7.3: The signal to noise ratio (SNR) derived from the region of interest signal mean and standard deviation. SNR response to increased measurement time, echo times and band width for the gradient echo sequence used to assess bone structure are shown. Profiles joining (●) and (▲) were recorded at a band width of 9.20 kHz while that joining (+) was recorded at a BW of 4 kHz.

images of the trabecular structure was obtained using a fast spoiled gradient echo pulse sequence (3D, TE=17.1 msec, TR=60 msec, θ=60°, BW=9.20 kHz). To improve the signal to noise ratio, three signal averages were prescribed. An axial localizer showing a region through which the high resolution image set was acquired is shown in figure 7.4. The region of interest defining the imaged volume was manually set around the centre of the radius as judged by the operator. The volume examined consisted of 12 contiguous slices. Each slice was acquired in the coronal plane with slice thickness of 0.8 mm, a 5 cm field of view, and a 256 square image matrix. This yielded an in-plane resolution of 195 μm x 195 μm x 800 μm. By acquiring 12 slices each with a thickness of 800 μm we examined approximately a 10 mm thick section of trabecular bone. The total scanning time required to acquire both the localizer scans and the high resolution image set was 20 minutes. A representative set of images is shown in figure 7.5.

7.2 Results of Pilot Study

A total of 8 female and 7 male volunteers underwent an MR examination to assess trabecular structure at the distal end of the radius. All were normal and without any diagnosed metabolic bone disease. Subject ages ranged from 24 to 72 years.

7.2.1 Intra-slice variability

To ensure that the same anatomic location was evaluated in all subjects, all indices of structure were calculated from 6 of the 12

Figure 7.4: A cross-sectional localizer image from which the volume to be imaged is set by the operator. The image is displayed in inverse grey scale so that bone appears white and fat black. A typical volume to be imaged for structure is displayed by the green box. This transaxial slice was abtained approximate 2 cm proximal to the growth plate at the distal end of the radius.

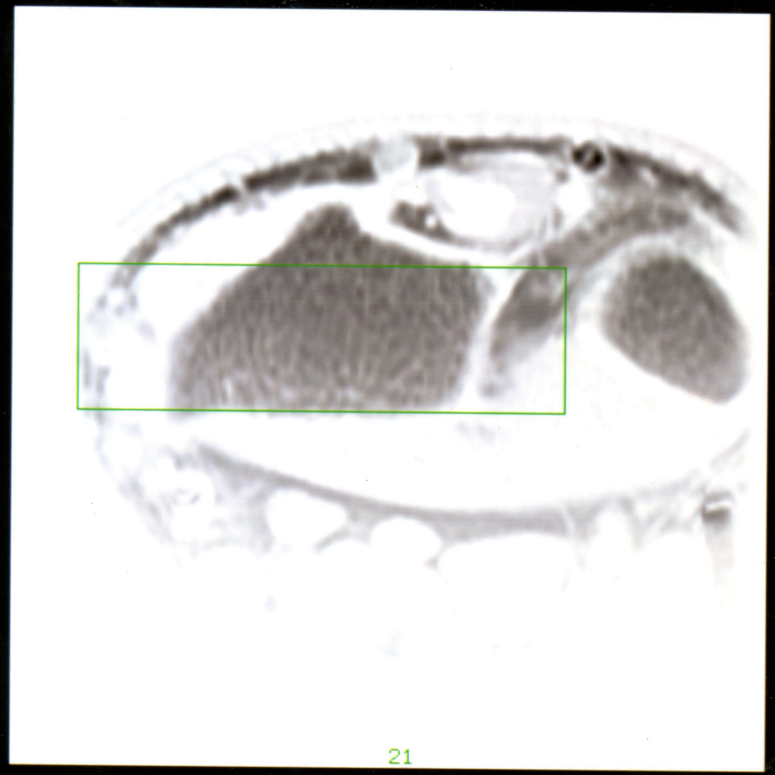
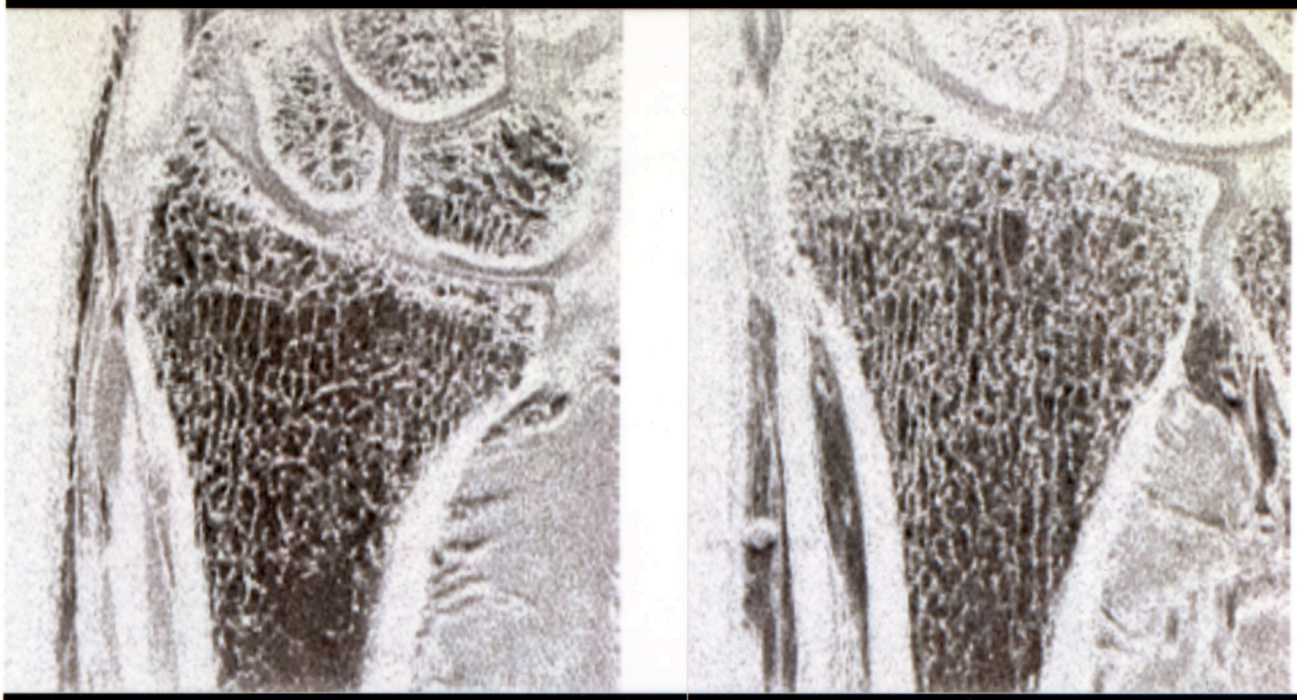


Figure 7.5: The trabecular bone structure at the distal radius of a 20 year old female volunteer (A) and 48 year old male volunteer (B). The images are displayed on an inverse grey scale so that the low bone signal appears white and the high signal from fat appears black.



A

B

slices obtained from the volume prescription. Using figure 7.4 as a guide these six slices were selected from the centre of the volume. That is, slices 1 to 3 and slices 10-12 were not examined. It is sensible to drop these slices from an assessment of structure because they may exhibit lower signal to noise and their location may be at a level of a transition between cortical and trabecular bone (subcortical bone). An average CI, H_A, and H_M was then calculated from the six estimates obtained. The variability in CI, H_A, and H_M among these six contiguous slices was assessed by deriving a coefficient of variation from the mean and standard deviation in CI, H_A, and H_M. This inter-slice variability was examined in four subjects. The results are summarized in table 7.2. These inter-slice variabilites suggest two observations about the bone structure in the radius. First, the structure changes little when examined as contiguous coronal images. Second, it is appropriate to quantitate the connectivity of the structure with a mean CI, H_A and H_M.

Another factor which contributes to the variability in the values of the indices derived from a given subject was the choice of the threshold used to correct the adaptive threshold step during segmentation. For example, when this threshold was varied from 40% to 60% of the maximum signal from fat, both CI and H_A varied with a coefficient of variation (CV) of less than 5%. The CV was equally as small for H_M and the integral gradient frequency (G_I).

7.2.1 Changes in connectivity with age

Table 7.2: The coefficient of variation (CV) for CI, H_A and H_M calculated from the six slices in each of four subjects examined.

Index	Mean CV(%)	Range of CV (%)
CI	7.8	5.1-11.4
H_A	8.8	5.7-10.7
H_M	9.1	5.9-12.3

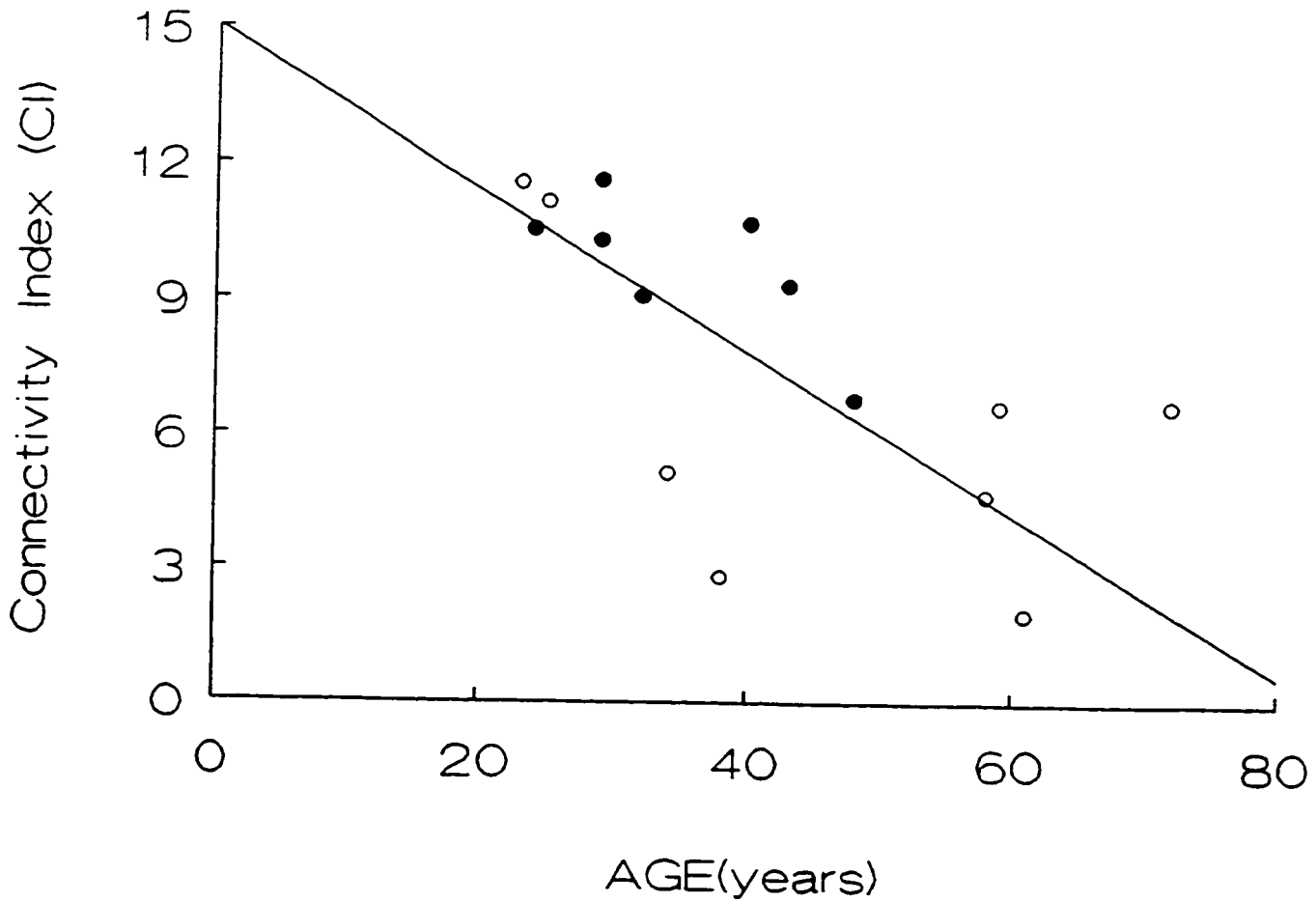


Figure 7.6: Relationship between age and the connectivity index (CI) derived from the distal end of the radius in 8 female (0) and 7 male (●) subjects. CI decreases at a rate of 0.14 per year ($r=-0.65$, $p<0.01$).

Figure 7.6 shows the age related change in the connectivity index derived from the 15 volunteers. Figures 7.7 and 7.8 show the same for the variables quantifying hole area. The data points plotted represent the mean calculated from the six most central slices in the volume imaged. The line plotted represents a least squares fit to the data points. As shown H_A and H_M increase at a rate of 0.014 $mm^2 year^{-1}$ and 0.46 $mm^2 year^{-1}$, respectively while CI decreases at a rate of 0.14 $year^{-1}$. With the limited number of subjects no conclusions could be drawn regarding gender differences. These indices change in a manner consistent with bone loss through aging. As previously noted, bone mass is lost by removal of entire trabeculae rather than by a generalized uniform thinning of the whole trabecular bone network. The resulting space left by lost trabeculae is filled by fatty marrow and remaining trabeculae are more widely separated and less connected. It is therefore not surprising that H_A and H_M increased with age while CI decreased.

7.2.3 Changes in orientation with age

Figure 7.9 shows the age related change in trabecular orientation from the 15 volunteers. Orientation is quantified by the integral gradient frequency G_I. Each point plotted represents the mean value calculated from the six slices analyzed. Clearly, there is no change in orientation with age (slope=-0.014, r=-0.17, p=0.55). This suggests that although the connectivity of the structure changes with age, the degree of anisotropy in the

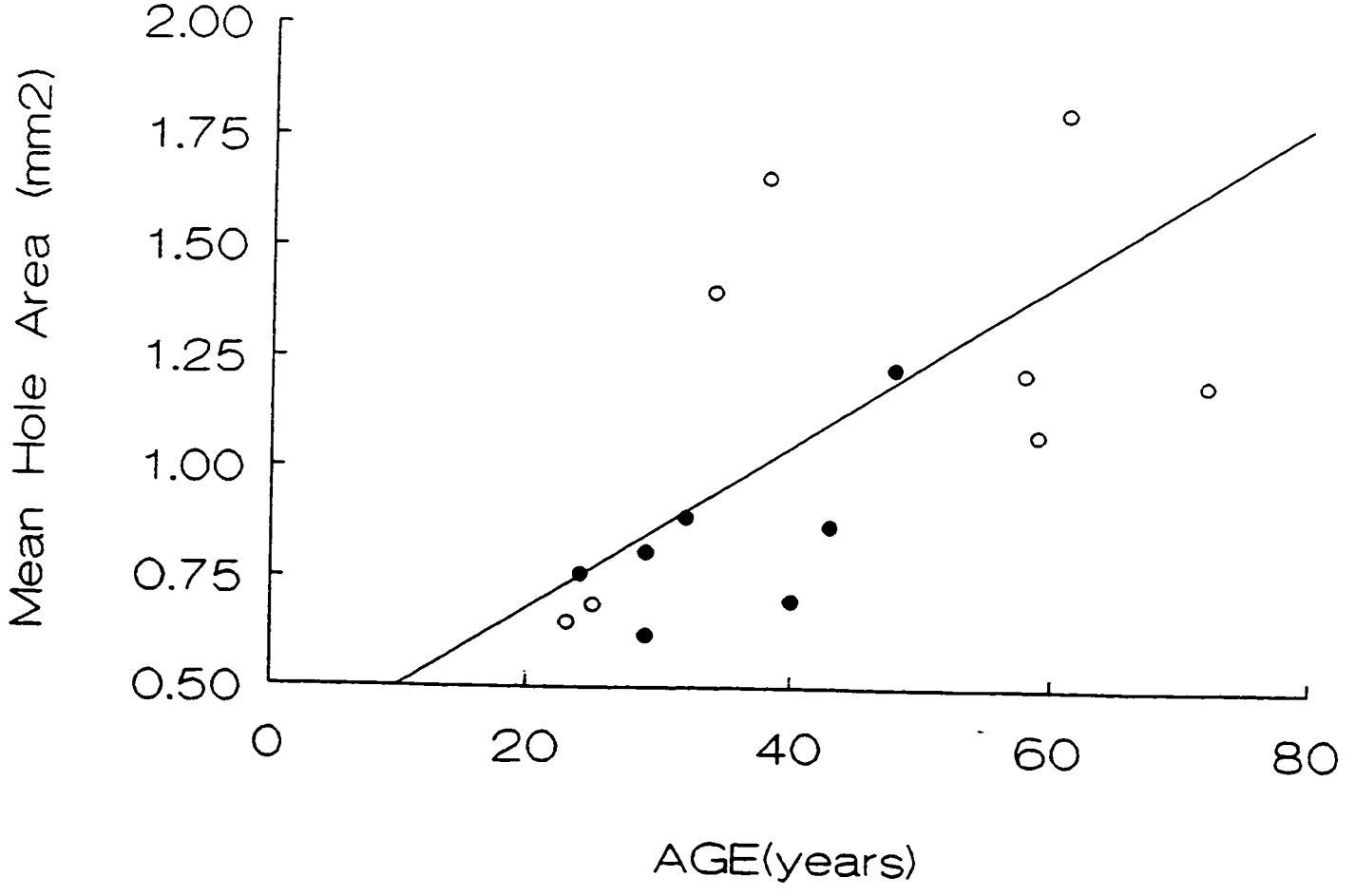


Figure 7.7: Relationship between age and the mean hole area (H_A) derived from the distal end of the radius in 8 female (0) and 7 male (●) subjects. H_A increases at a rate of 0.014 mm^2 per year ($r=0.59$, $p<0.02$).

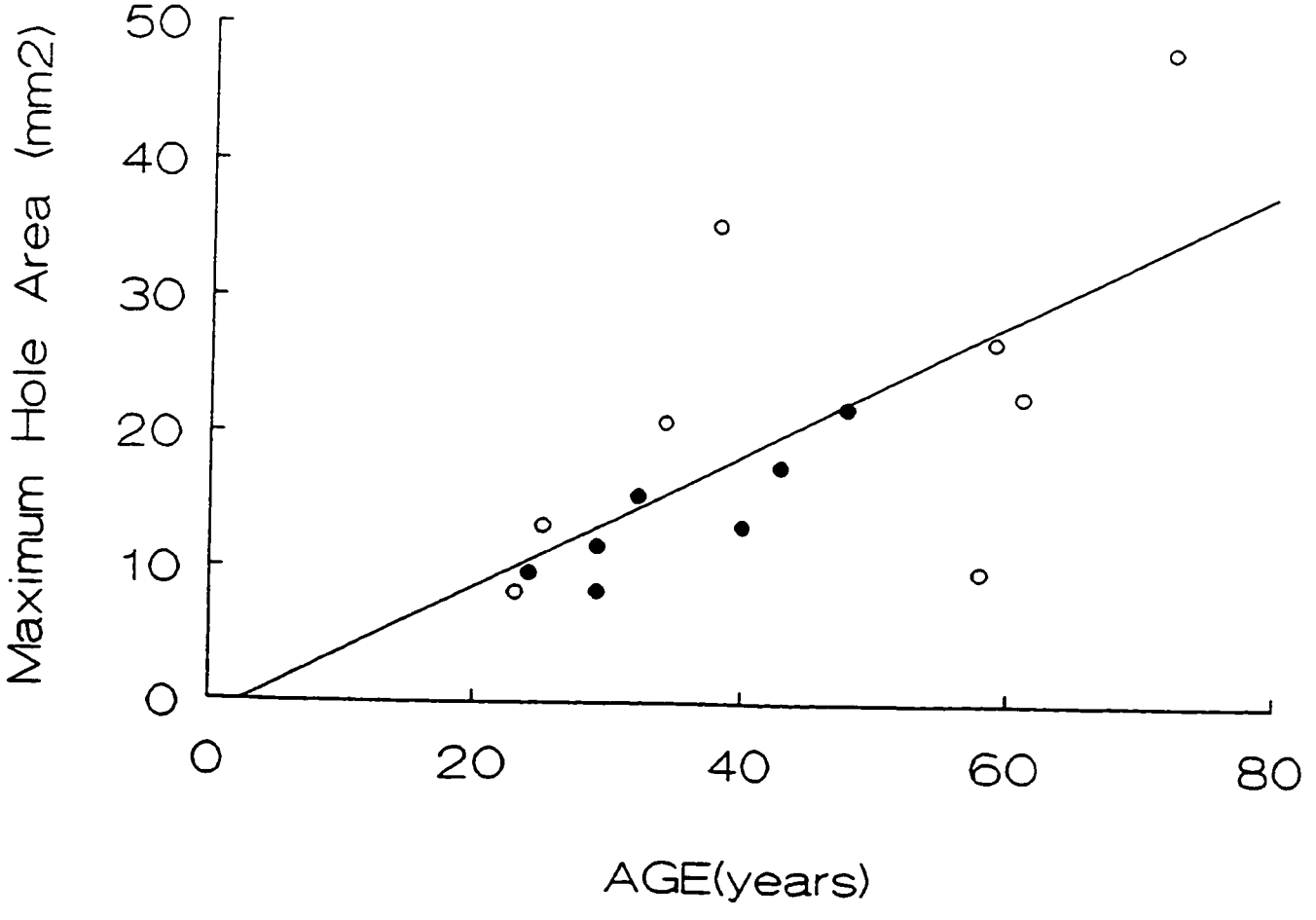


Figure 7.8: Relationship between age and the maximum hole area (H_M) derived from the distal end of the radius in 8 female (0) and 7 male (●) subjects. H_M increases at a rate of 0.46 mm^2 per year ($r=0.75$, $p<0.001$).

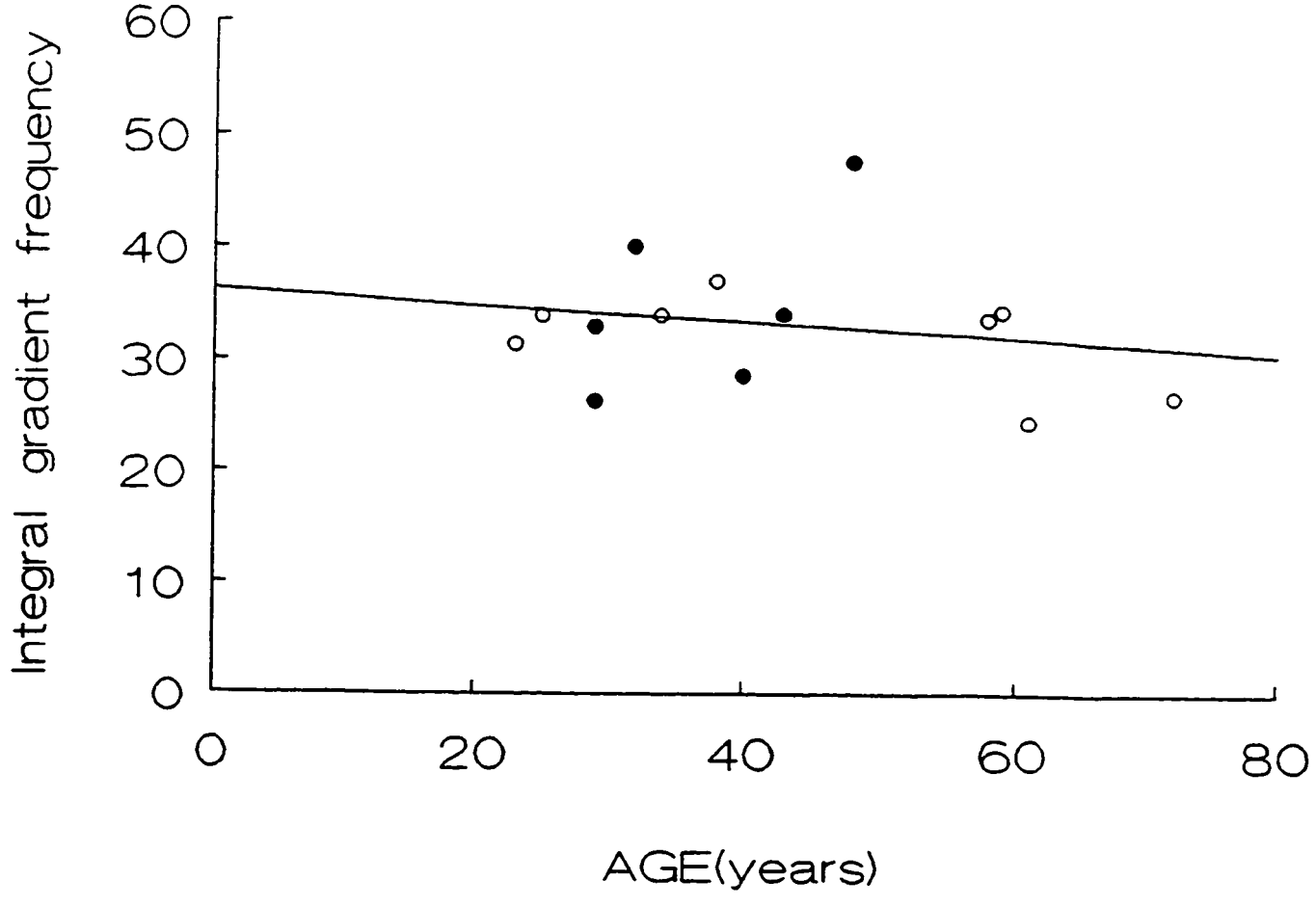


Figure 7.9: Relationship between age and trabecular orientation (G_I) derived from the distal end of the radius in 8 female (O) and 7 male (●) subjects. G_I does not change significantly with age ($r=-0.17$, $p<0.55$).

arrangement of the trabecular network remains unchanged.

7.3 Summary

In this study we used a standard clinical MR scanner to image trabecular bone at the distal end of the radius and developed an algorithm to segment the trabecular bone structure. From the segmented representation we were able to extract several indices from which the mechanical competence of the bone structure can be inferred. The significant rates of change in H_A, H_M and CI suggest that clinical magnetic resonance scanners may be sensitive enough to image small changes in trabecular bone architecture. One would expect that MR will be sensitive enough to discriminate the disrupted bone structure of persons with bone disease from those without. If so, MR can be used to study the response of the trabecular structure in such individuals to various therapeutic regimes.

Signal to noise considerations limit the acquisition of high resolution images of trabecular structure to peripheral sites such as the wrist. However, fractures of the distal radius have been shown to be a forecaster of subsequent hip fractures in the same individual (Mallmin et al., 1993). Therefore steps to improve the accuracy of imaging and segmentation of the trabecular architecture at the radius should be pursued.

Chapter 8

IN-VITRO ASSESSMENT OF TRABECULAR STRUCTURE: CORRELATION WITH STRESS TESTING

8.0 Introduction

The resistance of a bone to fracture can depend on bone mass, bone structure, bone geometry, and the presence of micro-damage. As demonstrated in the previous chapters, the structure of trabecular bone can be imaged in-vivo by MRI and CT and indices of structure derived from these images. The mechanical integrity of the imaged bone is then inferred from the derived indices. In this chapter, this inference is tested by quantifying the trabecular structure of isolated radii and relating it to predictions of bone strength derived from compressive testing.

8.1 Materials and Methods

8.1.1 Specimen description and preparation

Nine human cadaver radii were obtained from a commercial supplier (Osta International, White Rock, BC). All nine bones were intact but were defatted and therefore shipped in a dry state. The history, such as age, gender, and disease status of the individuals from which the specimens were obtained was unknown. However, 5 of the 9 specimens displayed various degrees of osteopenia, as judged qualitatively by radiographs.

8.1.2 Assessment of density

Each radius was scanned on an Hologic QDR 4500 (Hologic, Waltham, MA) using the forearm scanning software (version 8.11a:3). For scanning, each bone was placed on a tissue-

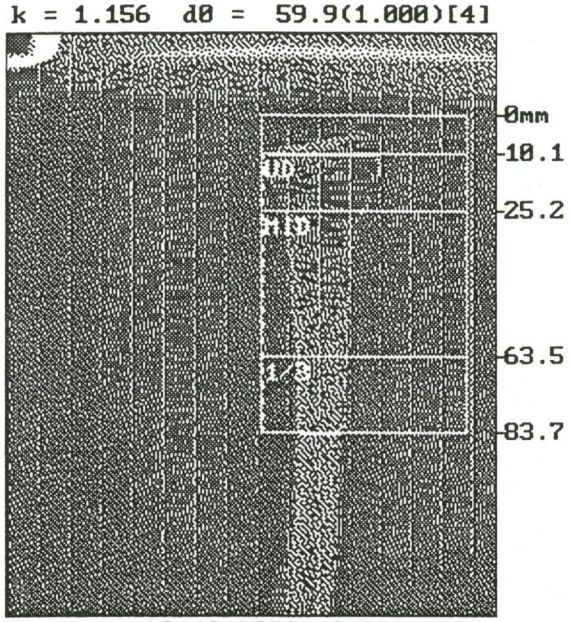
equivalent polyethylene plate approximately 1.5 cm thick. Each scan was analyzed for density utilizing the manufacturer suggested protocol. A typical analysis printout is given in figure 8.1. As shown, the automated evaluation software defines a global region of interest (total) and individual 1/3 distal (1/3), mid-distal (Mid), and ultra-distal (UD) regions of interest. The total bone density (BMD), bone mineral content (BMC), and projected area are reported for each of these regions of interest.

pQCT measurements were made on each of the nine bones. First, a coronal scout view of the radius was performed. Second, the axial slice of 2.5 mm thickness was performed at a region defined to be 5 mm proximal to the reference site indicated in figure 8.2. The pQCT image was automatically evaluated for trabecular bone density (TBD) and cortical bone density (CBD) with the system software. It is important to note that the location of the pQCT slice is within the UD region of interest defined by DXA. This UD region covers an area of 15 mm length over the distal radius along the long axis of the bone.

8.1.3 Assessment of structure

Each radius specimen was immersed in corn oil to simulate the yellow marrow normally present at the distal end of the radius. The oil was first heated and the immersed bones were evacuated to eliminate the air bubbles which may become trapped within the inter-trabecular spaces. High resolution MR images of each bone specimen were obtained in the axial plane using the in-vivo scanning protocol described in chapter 7. With this imaging

Chedoke-McMaster Hospitals



V0311961F Mon Mar 11 18:43 1996
Name: RADIUS TWO
Comment: chris gordon
I.D.: Sex: F
S.S.#: - - Ethnic:
ZIPCode: Height: 21.00 cm
Operator: Weight: kg
BirthDate: / / Age:
Physician:
Forearm Length: 22.2 cm
Image not for diagnostic use

TOTAL BMD CV IS LESS THAN 1.0%

C.F.	1.027	1.010	1.000

RADIUS	Area (cm2)	BMC (grams)	BMD (gms/cm2)
UD	3.55	0.98	0.276
MID	4.92	1.74	0.355
1/3	2.24	1.02	0.456
TOTAL	10.70	3.74	0.350

HOLOGIC

Figure 8.1: A typical bone mineral density summary report from DXA

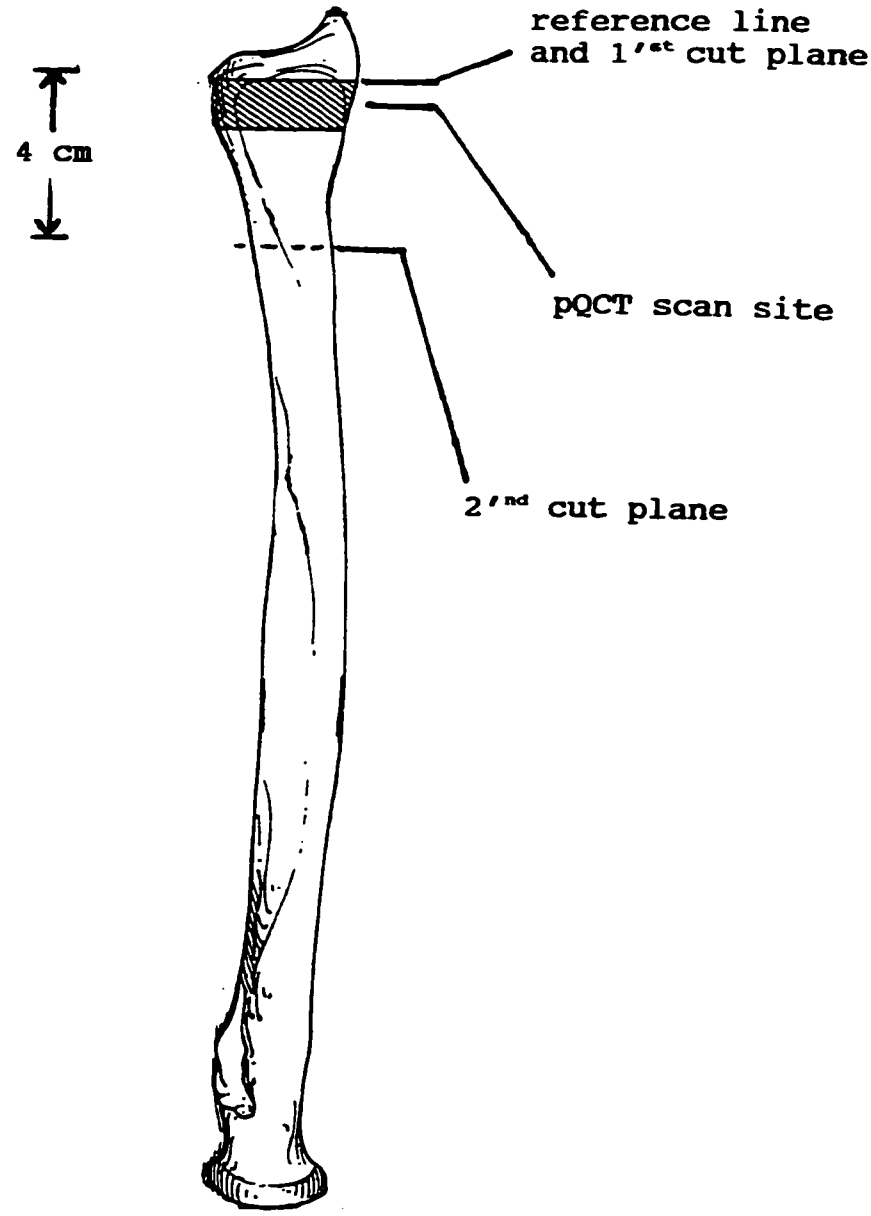


Figure 8.2: The basic geometry of an isolated radius. The site of the pQCT scan is indicated and the volume imaged by MR is given by the shaded area.

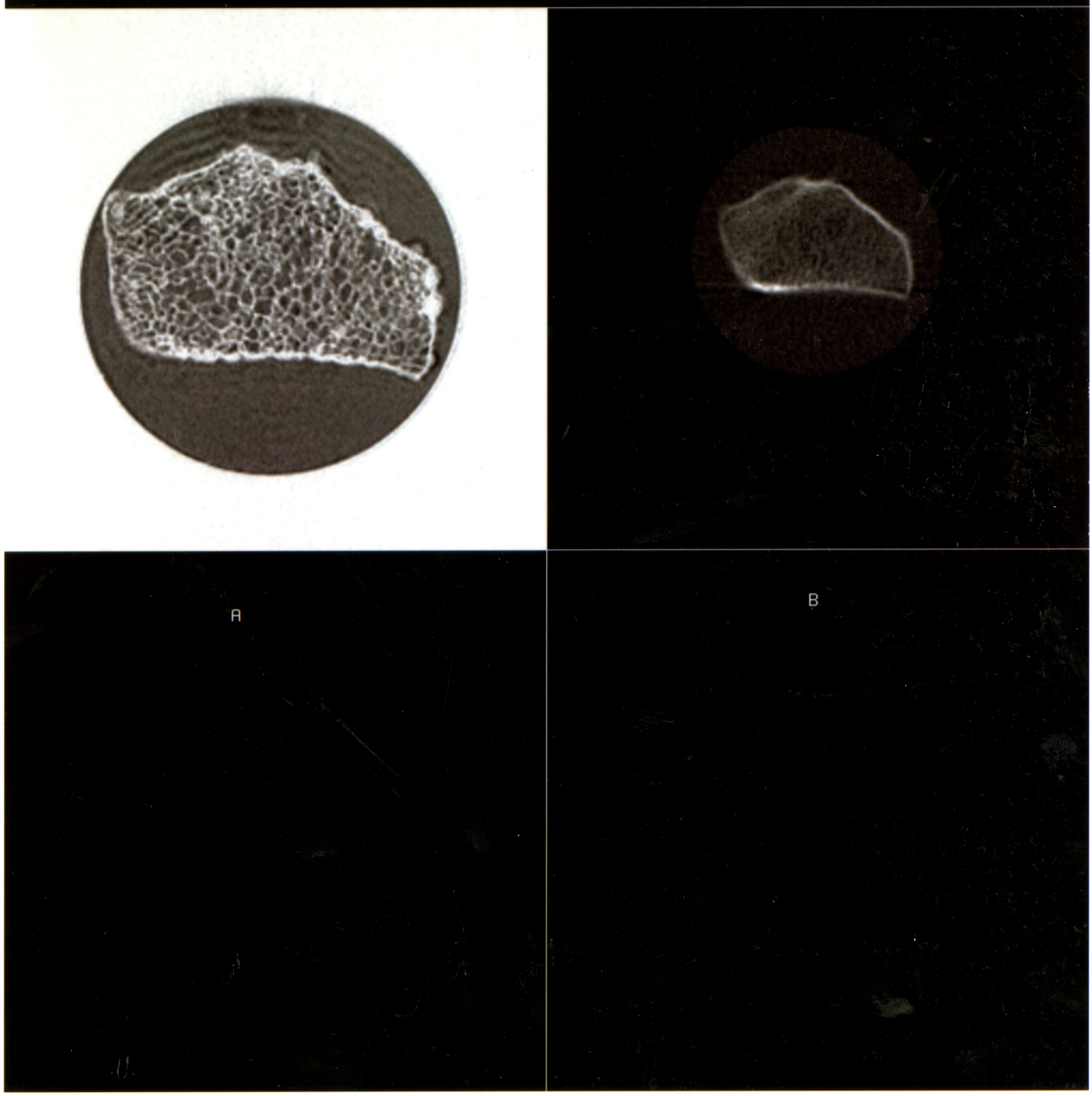
protocol, 12 contiguous images are obtained at a field of view of 5 cm, at 256 phase-encode and frequency-encode steps and with a slice thickness of 800 μm. This field of view and image matrix size yields an in-plane resolution of 195 μm. To improve the signal to noise ratio, 6 excitations were averaged. Therefore, the total imaging time for each specimen was approximately 30 minutes.

Both the MR and pQCT images were analyzed for structure using the postprocessing steps implemented for in-vivo assessments of structure. A typical MR image and pQCT image obtained at the same site is shown in figure 8.3. The MR image has been displayed in reverse grey scale to make its appearance compatible with the pQCT image. Trabecular bone was separated from the marrow background using thresholding, region growth, and skeletonization steps. From the processed image, marrow hole area and trabecular connectivity are quantified. Connectivity was assessed by the proposed connectivity index (CI) and the marrow space was quantified by a mean hole area (H_A) and a maximum hole area (H_M).

8.1.4 Mechanical testing

The maximum compressive strength of each radius specimen was determined using a Lloyd's material testing unit (Lloyd's Instruments, Fareham, UK). The specimens were first prepared for crushing by cutting two plano-parallel ends which were 4 cm apart. The site of each cut is indicated in figure 8.2. After preparation, each sample was placed in the press and compressed at a strain rate of 3.0 cm per minute. This strain

Figure 8.3: Comparison of an MR image (A) and a pQCT image (B) recorded at the distal end of the radius. Notice that the cortical shell at this scan site is very thin.



rate is consistent with that imparted to the radius following a fall from a standing height (Sparado et al 1994). A diagram of the setup for the compressive test is given in figure 8.4. The load data is acquired directly by a personal computer interfaced to a load cell. The displacement data is saved into a spreadsheet and manipulated to yield a typical displacement curve as shown in figure 8.5 for one of the 9 specimens. Two points are worth highlighting from this plot. First, in this specimen the maximum compressive strength was 3484 N. Second, after the bone fractures, the load drops dramatically and the test was halted.

8.1.5 Data analysis

All statistical analyses were performed with Statistix (version 4.0). Linear correlation analyses were used to assess the strength of the relationships between indices of density, trabecular bone structure, and the peak load at fracture. The significance of each correlation was estimated with the use of the t-test. The additional or incremental contribution of trabecular bone structure parameters (in addition to trabecular bone density) to the prediction of the peak load was examined using a linear multiple regression model.

8.2 Results

Peak fracture load, along with the indices of density and cross-sectional area derived by pQCT and DXA in the 9 specimans are summarized in table 8.1. Although this study was conducted on isolated radii the peak fracture loads listed fall within the

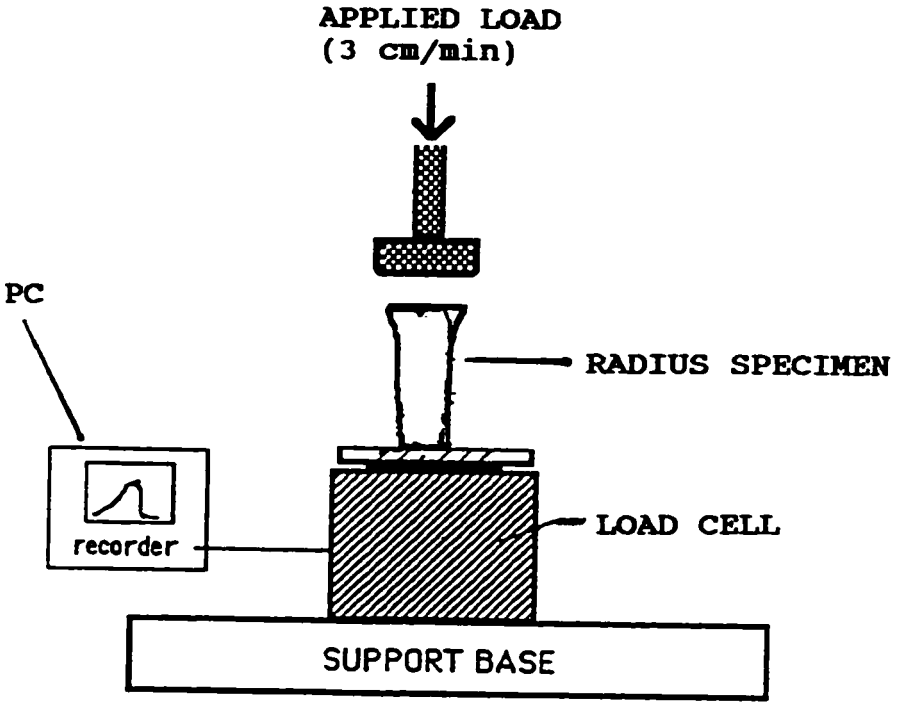


Figure 8.4: Diagram of the setup for compressive fracture testing. Loading is increased and data on load deformation is recorded via a personal computer (PC) interfaced to the load cell.

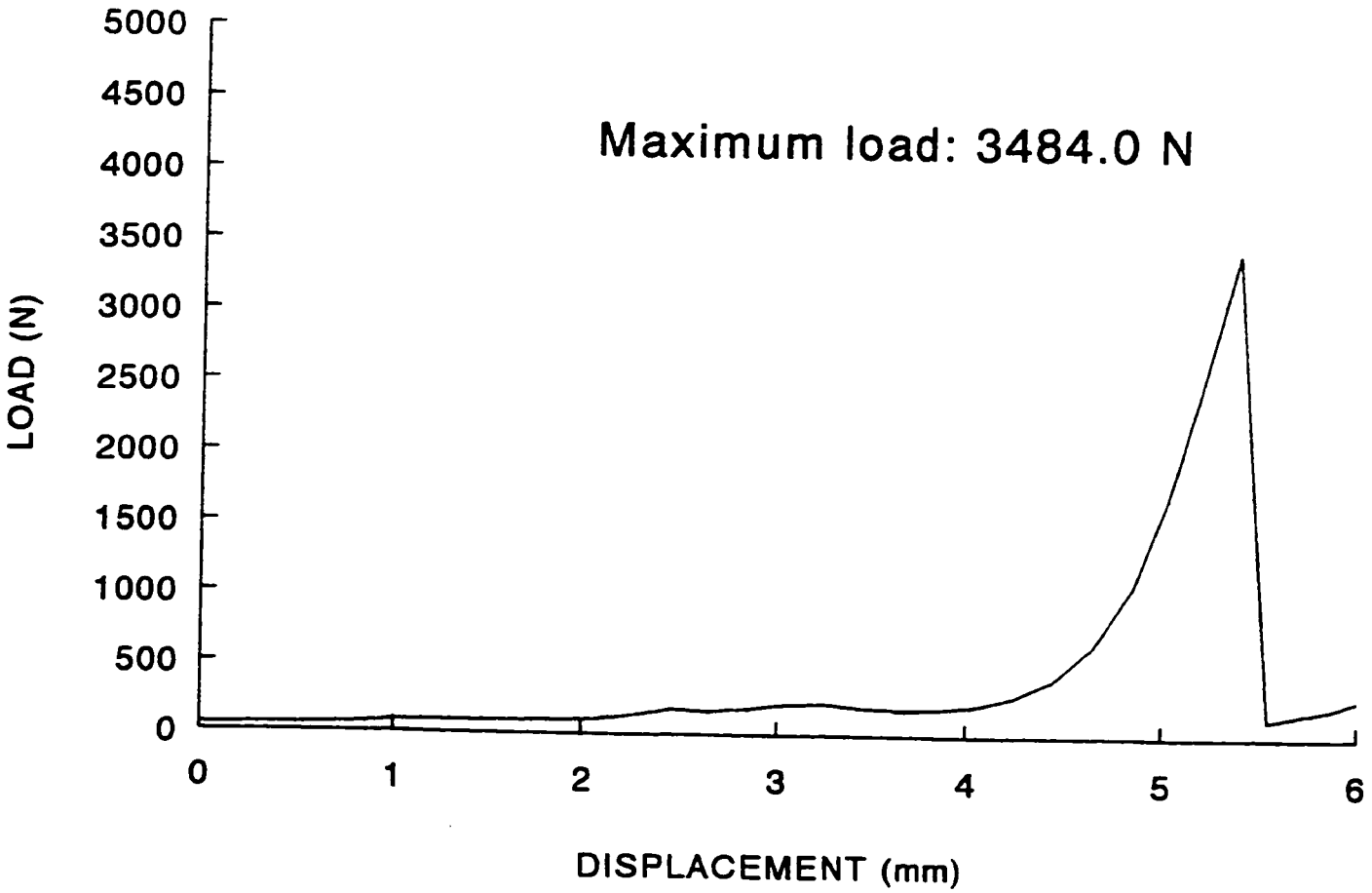


Figure 8.5: A typical displacement curve produced by the compressive test.

Table 8.1: The peak load at fracture and densitometric measures recorded in the 9 radius specimens tested.

Bone (#)	Load (N)	CBA (mm^2)	CBD (mg cm^{-3})	TBA (mm^2)	TBD (mg cm^{-3})	UD (g cm^{-2})	MID (g cm^{-2})
1	2756.0	96.4	545.9	243.7	220.3	0.384	0.556
2	1119.0	104.1	313.0	256.1	141.0	0.276	0.355
3	3484.0	98.4	408.4	335.1	168.8	0.343	0.338
4	1270.0	82.6	414.6	179.8	130.5	0.285	0.322
5	922.4	99.8	493.8	262.7	159.1	0.385	0.440
6	4726.0	96.9	611.4	242.4	197.0	0.437	0.521
7	1511.0	130.6	313.6	353.0	139.6	0.325	0.399
8	645.4	115.0	268.5	309.0	93.4	0.258	0.376
9	1202.0	87.7	339.9	204.3	153.1	0.277	0.335

Load= peak load at fracture
CBA= cortical bone area from pQCT
CBD= cortical bone density from pQCT
TBA= trabecular bone area from pQCT
TBD= trabecular bone density from pQCT
UD= ultra-distal bone density from DXA
MID= mid-distal bone density from DXA

range of values reported by others who tested intact cadaver specimens. For example, Myers et al (1991) reported that a mean value of 3390 N produced radius fractures in the specimens tested. Another investigation of peak fracture loads reported lower values (1640 N) (Sparado et al 1994).

The trabecular structure parameters derived from MR and pQCT images and peak fracture loads characterizing each specimen are shown in table 8.2. The correlation coefficients relating fracture load with densitometric, cross-sectional area, and structural parameters determined from both pQCT and MRI are given in table 8.3. As indicated, measures of density were moderately good indicators of strength. The best correlation with fracture load was for UD (r^2=0.57) which reflects the combined density of cortical and trabecular components. Surprisingly, measures of bone size such as CBA and TBA were very poor (r^2<0.1) indicators of bone strength. In contrast to the significant coefficients of determination associated with density measures, correlations between structural parameters and fracture load were weaker. Only the connectivity index derived from pQCT images correlated significantly with peak load (r^2=0.49, n=9, p<0.04).

The linear correlations relating trabecular bone density and structural parameters derived from pQCT are given in table 8.4. The inter-correlations between structure assessed by MR and pQCT are also listed in this table. Despite differences in slice thickness and in-plane resolution, indices of structure estimated from pQCT showed moderate to good correlation with those derived from MR (r^2>0.6). This suggests that information relating to trabecular structure can be extracted from images which are

Table 8.2: The peak load at fracture and structural parameters recorded by MR and pQCT in the 9 radius specimens tested.

Bone (#)	Load (N)	CI_CT	H_A_CT (mm^2)	H_M_CT (mm^2)	CI_MR	H_A_MR (mm^2)	H_M_MR (mm^2)
1	2756.0	22.56	0.58	37.13	13.02	0.65	4.49
2	1119.0	19.19	0.61	12.85	9.24	0.96	7.07
3	3484.0	19.60	0.74	40.40	11.81	0.79	6.31
4	1270.0	16.22	0.67	19.60	9.33	1.04	15.02
5	922.4	17.56	0.65	19.06	8.08	1.23	30.08
6	4726.0	23.58	0.53	36.26	10.30	0.79	15.86
7	1511.0	18.03	0.83	41.60	8.45	1.17	15.51
8	645.4	8.19	1.69	81.89	3.11	2.61	138.56
9	1202.0	19.00	0.61	15.25	5.17	1.41	29.81

Load= peak load at fracture.
CI_CT= connectivity index from pQCT.
H_A_CT= mean hole area from pQCT.
H_M_CT= maximum hole area from pQCT.
CI_MR= connectivity index from MR.
H_A_MR= mean hole area from MR.
H_M_MR= maximum hole area from MR.

Table 8.3: Correlation with load for the densitometric and structural parameters used to characterize the bone specimens. Statistical significance at the $p<0.05$ level is indicated by (*).

Variable	r^2	p-value	
CBA	0.034	0.636	
CBD	0.520	0.028	*
TBA	0.005	0.858	
TBD	0.538	0.024	*
UD	0.569	0.019	*
MID	0.280	0.143	
CI_CT	0.491	0.036	*
H_A_CT	0.173	0.265	
H_M_CT	0.002	0.918	
CI_MR	0.435	0.053	
H_A_MR	0.362	0.086	
H_M_MR	0.188	0.241	

Table 8.4: Correlation of structural parameters assessed by MRI and pQCT. The correlation of trabecular density and the structural parameters derived from pQCT are also given. R^2 values are listed and statistical significance at the $p<0.05$ level is indicated by (*).

	TBD	CI_MR	H_A_MR	H_M_MR
CI_CT	0.810*	0.606*	N/A	N/A
H_A_CT	-0.518*	N/A	0.831*	N/A
H_M_CT	-0.109	N/A	N/A	0.594*

limited by the effects of volume averaging. The correlations listed in table 8.4 also indicate that both CI and H_A are moderate indicators of trabecular bone density.

To determine whether a combination of structural parameters added significant information to the prediction of fracture load, multiple linear regression models were fitted with peak load as the outcome variable. The coefficients of determination, r^2, are given in table 8.5 for models based on structural parameters alone. Results are also given for models in which trabecular bone density and a combination of one or more structural parameters were the independent variables. Note that trabecular bone density was combined only with structural parameters derived from pQCT. The inclusion of a single structural parameter with TBD did not significantly increase the prediction of bone strength beyond that offered by TBD alone. The r^2 for TBD and load is included in this table for comparison.

A combination of structural parameters clearly improved the prediction of peak load at fracture. When combined, variables relating to trabecular spacing (namely H_A and H_M) made a significant contribution to the prediction of load. This was the case whether H_A and H_M were estimated by pQCT (r^2=0.82, $(r^2)_{adj}$=0.76) or by MRI (r^2=0.7, $(r^2)_{adj}$=0.60). Although the combination of TBD, H_A_CT, and H_M_CT produced the largest correlation with load, this multivariate fit to the data must be considered with caution given the limited sample size. However, this result illustrates the potential of these structural variables to add significant information to bone mineral density in the prediction of peak fracture loads.

Table 8.5: Predictors of peak load at fracture when densitometric and structural parameters are examined using multiple linear regression analysis. Statistical significance at the $p<0.05$ level is indicated by (*).

Variables	r^2	$(r^2)_{adj}$	p-value	
TBD	0.538	****	0.024	*
(TBD,CI_CT)	0.547	0.396	0.093	
(TBD,H_A_CT)	0.564	0.419	0.083	
(TBD,H_M_CT)	0.628	0.504	0.051	
(CI_CT,H_A_CT)	0.744	0.658	0.017	*
(CI_CT,H_M_CT)	0.805	0.740	0.007	*
(H_A_CT,H_M_CT)	0.821	0.761	0.006	*
(TBD,H_A_CT,H_M_CT)	0.834	0.734	0.021	*
(CI_MR,H_A_MR)	0.435	0.246	0.181	
(CI_MR,H_M_MR)	0.461	0.281	0.157	
(H_A_MR,H_M_MR)	0.697	0.595	0.028	*

It is recognized that in this study only a small number of radius specimens were tested. However, the results corresponding to the hole size variables H_A and H_M are extremely encouraging. The results of the multiple linear regression combining H_A and H_M derived from pQCT and MRI images are given below in equations 8.1 and 8.2, respectively.

$$\text{Load}=3825-7048(H_A\_CT)+105(H_M\_CT) \quad (r^2=0.82,\ (r^2)_{adj}=0.76,\ n=9,\ F=13.75,\ p=0.006) \qquad 8.1$$

$$\text{Load}=7565-6569(H_A\_MR)+74(H_M\_MR) \quad (r^2=0.70,\ (r^2)_{adj}=0.60,\ n=9,\ F=6.89,\ p=0.03) \qquad 8.2$$

Comparing these two equations, it is interesting that the weighting coefficients are not significantly different. This is an important observation because it says that, although examined with two distinctly different imaging modalities, a similar conclusion can be drawn about trabecular bone structure in the nine bones tested. Simply put, a combination of the variables relating the size of the marrow space may be the best predictor of peak fracture loads. To emphasize this point, the regression equation given by equation 8.1 is plotted in figure 8.6. An increase in marrow space requires less load to fracture. This is not surprising given what is known about bone loss through aging. Normal loss of trabecular bone is mainly due to entire trabeculae being removed. The remaining trabecular network is less connected, contains larger marrow spaces and is less likely to withstand a compressive force.

8.3 Summary

Although limited by sample size, this study yielded some

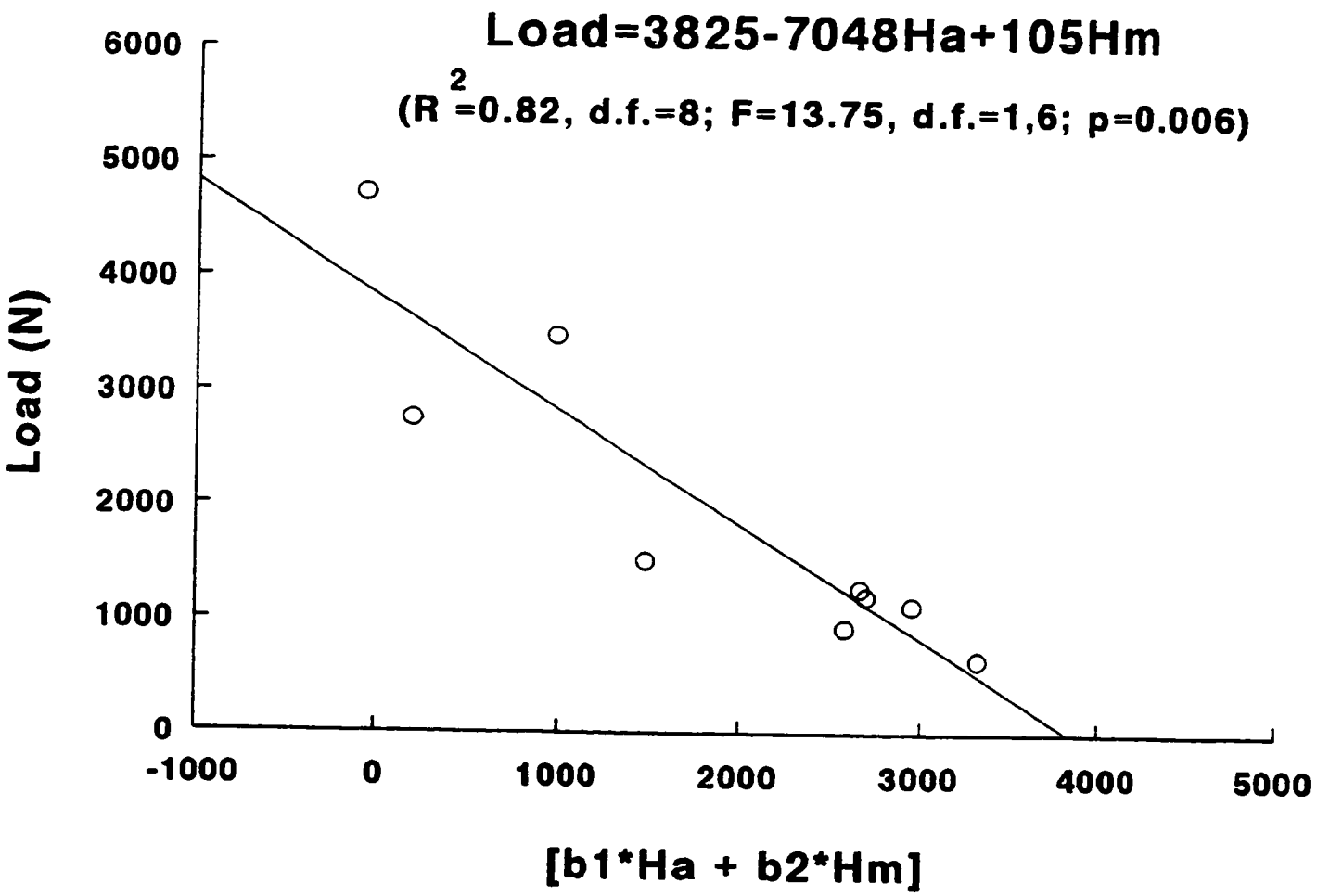


Figure 8.6: The regression of load against the combination of hole size variables.

encouraging results. It appears that indices of trabecular structure are important determinants of fracture load along with bone mineral density. In seeking the features that were the most reliable indicators of bone strength at the distal end of the radius, a combination of the mean hole area and maximum hole area had the highest correlation with load. This held true whether these two variables were derived from pQCT or MR images.

Chapter 9

CONCLUSION

Normal bone mass may not be sufficient to prevent fracture but low bone mass is not inevitably associated with fracture. Variations in trabecular bone architecture are thought to be a major factor which might contribute significantly to fracture risk. This work has examined ways of assessing trabecular bone structure at the distal end of the radius in-vivo to better understand the contribution of architecture to fracture risk. To this end, it proceeded on four major fronts. First, images of sufficient resolution were acquired using a commercial pQCT scanner and a clinical MR imager. Second, the image processing software necessary to segment the imaged trabecular structure was developed. This software was implemented in C for a UNIX platform. Third, two indices were proposed to quantify the connectivity of the segmented structure. One index was derived from the application of trabecular strut analysis to a skeletonized representation of the bone network. The other quantified the marrow space by deriving a mean hole area and maximum hole area of the bone structure as it appears in two dimensions. The clinical value of these indices was tested by conducting pilot studies which examined the ability of the indices to discriminate a small group of Colles fracture patients from the normal population and to reflect normal age related changes in structure. The fourth and last stage of this work

examined the proportion of the variance in compressive strength of a group of radius bones that can be accounted for by bone mineral density and bone architecture. A summary of the results derived from each stage of this work follows. A full listing of the peer reviewed publications and abstracts generated from this work is given in appendix A.

9.1 In-vivo assessment of trabecular structure with pQCT

A dedicated computed tomography system was used to acquire transaxial images of the distal radius to assess trabecular bone structure in-vivo. The CT images were acquired with a Stratec XCT 960 scanner (Norland Corporation, Wisconsin). This is a special purpose, second generation, peripheral Quantitative Computed Tomography (pQCT) scanner used to acquire transaxial images of the distal radius. Images were recorded with a slice thickness of 2.5 mm and reconstructed onto a 256 square matrix with a pixel size of 0.33 mm. This pixel size was just sufficient to allow trabecular structure to be visualized. Trabecular bone was segmented from the marrow and soft tissue background by postprocessing the image with a region grow and skeletonization step. From the processed image the integrity of the bone was assessed by examining the continuity of its trabecular network and by determining the area of the holes comprising its marrow space. The continuity of the bone imaged was assessed by a proposed connectivity index (CI) and the size of the marrow spaces was assessed by calculating a mean hole area (H_A) in the bone cross-section. Repeat measurements revealed that the

intra-subject variability in CI and H_A was small (CV<6%). Both CI and H_A were sensitive enough to reflect differences in structure at the head of the radius and at several sites along its shaft.

The diagnostic value of assessing bone structure at the distal end of the radius was tested by measuring trabecular bone density, CI and H_A in a mixed group of 44 subjects, 21 of whom had suffered a wrist fracture. It was found that a trabecular bone density threshold of 138 mg/cm^3, corresponding to 2 standard deviations below the mean density in the 23 non-fractured subjects separated fractured from non-fractured subjects with a sensitivity of 38% and a specificity of 100%. A CI threshold of -7.6 increased the sensitivity (48%) and maintained the high degree of specificity (96%). An H_A threshold of 4.4 mm^2 achieved a sensitivity of 67% and a specificity of 96%. This increased sensitivity achieved by the proposed indices suggests that an in-vivo assessment of trabecular bone structure can contribute significantly to the identification of persons at risk of fracture.

9.2 In-vivo assessment of trabecular structure with MRI

In this work a protocol for assessing trabecular bone structure at the distal end of the radius from high-resolution magnetic resonance images was established. MR images were acquired on a 1.5T General Electric Signa clinical imager. A three dimensional gradient echo pulse sequence used in conjunction with a surface coil yields high resolution images in each of the three orthogonal planes and at a voxel size of 192 X 192 X 800 μm.

Trabecular bone was again segmented from the marrow and soft tissue background using thresholding, region growth, and a skeletonization step. From the segmented image the connectivity and orientation of the trabecular bone network was measured. Connectivity was assessed by the proposed connectivity index and marrow space was quantitated by a mean hole area and a maximum hole area (H_M). Significant age-related changes in CI and H_A were observed in a mixed group of normal volunteers. CI decreased at a rate of 0.14 $year^{-1}$ ($r=0.65$, $n=15$, $p<0.01$) and H_A and H_M increased at rates of 0.014 $mm^2 year^{-1}$ ($r=0.59$, $n=15$, $p<0.02$) and 0.46 $mm^2 year^{-1}$ ($r=0.75$, $n=15$, $p<0.001$), respectively. Gradient analysis was used to examine trabecular orientation, and revealed that the individual trabeculae at the distal end of the radius are organized anisotropically along the bone. These findings suggest that clinical magnetic resonance scanners can be used to assess trabecular bone structure in-vivo.

9.3 In-vitro structure: correlation with stress testing

The mechanical integrity of the imaged bone was inferred from the structural indices derived. This work concluded by testing this inference by quantifying structure in a set of isolated radii and relating it to predictions of bone strength based on mechanical testing. To this end, trabecular structure at the distal end of 9 cadaveric radii was imaged by CT and MRI. Trabecular bone was segmented from fat and the indices relating to network connectivity and the size of the marrow space were derived. Bone density of each radius was also assessed by dual energy x-ray absorptiometry. Each

bone was subjected to a mechanical load consistent with a fall from a standing height and density, CI, H_A and H_M were compared to the compressive strength.

The results obtained indicated that, along with density, an increase in CI and a decrease in H_A and H_M better maintain the mechanical integrity of trabecular bone at the distal end of the radius. To determine whether the structural parameters added significant information to the prediction of peak fracture load, multiple linear regression models were fitted with peak load as the outcome variable. The coefficient of determination, r^2, was less than 0.6 when bone mineral density alone was included in the model. The combination of H_A and H_M made a significant contribution to the prediction of peak load. This was true whether H_A and H_M were derived from pQCT ($(r^2)_{adj}$=0.76) or MRI ($(r^2)_{adj}$=0.60) images.

9.4 Conclusion and future work

There are several clinical implications which may be drawn from this work. First, it must be noted that Colles' fracture (a fracture of the distal 3 cm of the radius) is the most common fracture in women less that 75 years old in the United States (Owen et al 1982) and Northern Europe (Alffram et al 1962, Solgaard and Petersen 1985), and particularly past the age of 40 years there is a sharp increase in the prevalence of fractures of the distal radius. This suggests that, with advancing age and bone loss, less force is required to cause a fracture during a fall. When compared with age-matched control subjects, studies of patients with a

Colles fracture have found relatively small reductions in bone mineral density at the distal radius which restricts the degree to which these subjects can be discriminated from the normal population. (Harma and Karjalainen 1986, Eastell et al 1989, Gardsell et al 1989).

In contrast, the structural parameters proposed in this work better discriminated Colles fracture patients than did measures of bone mineral density (Gordon et al 1996). Furthermore, these parameters were sufficiently sensitive to detect age related changes in trabecular architecture (Gordon et al 1997). Therefore, these structural indices may represent a potentially exciting and promising means of discriminating fracture outcomes and monitoring normal changes in trabecular bone structure. This potential should be explored at other clinically relevant sites such as the lumbar spine and the hip.

To further the assessment of trabecular architecture the possibility of obtaining three dimensional architectural information from contiguous images should be explored. Exploring such avenues will advance the possibility of routine in-vivo assessments of trabecular architecture into the clinical arena.

REFERENCES

Ackerman JL, Raleigh DP, and Glimcher MJ. Phosphorus-31 magnetic resonance imaging of hydroxyapatite: A model for bone imaging. Magn Res Med, 1992;25:1-11.

Alffram PA and Bauer GC. Epidemiology of fractures of the forearm: a biomechanical investigation of bone strength. J Bone Jt Surg, 1962;44:105-114.

Amstutz HC and Sissons HA. The structure of the vertebral spongiosa. J Bone Jt Surg, 1969;51B:540-550.

Argren M, Karellas A, Leahey D, Marks S, and Baran D. Ultrasound atenuation of the calcaneus: a sensitive and specific descriminator od osteipenia in postmenopausal women. Calcif Tissue Int, 1991;48:240-244.

Arnold JS. Amount and quality of trabecular bone in osteoporotic vertebral fractures. J Clin Endocrinol Metab, 1973;2:221-238.

Birkenhager-Frenkel DH, Courpron P, Hupscher EA, Clermonts E, Coutinho MF, Schmitz PI, and Meunier PJ. Age-related changes in cancellous bone structure: A two-dimensional study in the transiliac and iliac crest biopsy sites. Bone and mineral, 1988;4:197-216.

Boone JM and Seibert A. An analytical edge spread function for computer fitting and subsequent calculation of the LSF and MTF. Med Phys, 1994;21:1541-1545.

Brandenburger GH. Clinical determination of bone quality: is ultrasound the answer?. Calcif Tissue Int, 1993;53(S1):S151-S156.

Brooks RA and Di Chiro G. Principles of computer assisted tomography (CAT) in radiographic and radioisotope imaging. Phys Med Biol, 1976;21:689-732.

Caldwell B, Willett K, Cuncins AV, and Hearn TC. Characterization of vertebral strength using digital radiologic analysis of bone structure. Med Phys, 1995;22:611-615.

Cann CE and Genant HK. Precise measurments of vertebral mineral content using Computed Tomography. J Comput Assist Tomogr, 1980;4:493-500.

Cann CE, Genant HK, Kolb FO, and Ettinger B. Quantitative computed tomography for prediction of vertebral fracture risk. Bone, 1985;6:1-7.

Chalmers J and Weaver JK. Cancellous bone: Its strength and changes with aging and an evaluation of some methods for measuring its mineral content. J Bone Jt Surg, 1966;48A:299-308.

Chevalier F, Laval-Jeantet AM, Laval-Jeantet M, and Bergot C. CT image analysis of the vertebral trabecular network in-vivo. Calcif Tissue Int, 1992;51:8-13.

Ciarelli MJ, Goldstein SA, Kuhn JL, Cody DD, and Brown MB. Evaluation of orthogonal properties and density of human trabecular bone from the major metaphyseal regions with materials testing and computed tomography. J Orthop Res, 1991;9:674-682.

Compston JE, Garrahan NJ, Croucher PI, Wright CP, and Yamaguchi K. Quantitative analysis of trabecular bone structure. Bone, 1993;14:187-192.

Cummings SR. Are patients with hip fractures more osteoporotic?: Review of the evidence. Am J Med, 1985;78:487-494.

Davis CA, Genant HK, and Dunham JS. The effects of bone on proton NMR relaxation times of surrounding liquids. Invest Radiol, 1986;21:472-477.

Davis GR and Wong L. X-ray microtomography of bones and teeth. Physiol Meas, 1996;17:121-146.

Dooms GC, Hricak H, Margulis AR, and De Greer G. MR imaging of fat. Radiology, 1986;158:51-58.

Durand EP and Ruegsegger P. Cancellous bone structure: Analysis of high resolution CT images with the run-length method. J Comput Assist Tomogr, 1991;15:133-139.

Eastell R, Wahner HW, O'Fallon M, Amadio PC, Melton JL, and Riggs LB. Unequal decrease in bone density of lumbar spine and ultradistal radius in Colles' and vertebral fracture syndromes. J Clin Invest, 1989;83:168-174.

Feldkamp LA, Goldstein SA, Parfitt AM, Jesion G, and Kleerekoper M. The direct examination of three-dimensional bone architecture in vitro by computed tomography. J Bone Miner Res, 1989;4:3-11.

Foo TK, Shellock FG, Hayes CE, Schenck JF, and Slayman BE. High-resolution MR imaging of the wrist and eye with short TR, short TE, and partial-echo acquisition. Radiology, 1992;183:277-281.

Ford JC and Wehrli FW. In-vivo quantitative characterization of trabecular bone by NMR interferometry and localized proton spectroscopy. Mag Res Med, 1991;17:543-551.

Fullerton GD. Basic concepts for nuclear magnetic resonance imaging. Magnetic Resonance Imaging, 1982;1:39-55.

Gardsell P, Johnell O, and Nilsson BE. Predicting fractures in women by using forearm bone density. Calcif Tissue Int, 1989;44:235-242.

Genant HK, Steiger P, Block JE, and Gluer CC. Quantitative computed tomography: Update 1987. Calcif Tiss Int, 1987;41:179-186.

Geraets W, Van der Stelt P, Netelenbos CJ, and Elders PM. A new method for automatic recognition of the radiographic trabecular pattern. J Bone Miner Res, 1990;5:227-233.

Gluer CC, Steiger P, Selvidge R, Elliesen-Kliefoth K, Hayashi C, and Genant HK. Comparative assessment of dual-photon absorptiometry and dual-energy-radiography. Radiology, 1990;174:233-228.

Goldstein SA. The mechanical properties of trabecular bone: Dependence on anatomic location and function. J Biomechanics, 1987;20:1055-1061.

Gonzalez WC and Woods RE. Digital image processing. (Reading:Addison-Wesley) ch 4&7, 1992.

Gordon CL, Webber CE, Adachi JD, and Christoforou N. In-vivo assessment of trabecular bone structure at the distal radius from high-resolution computed tomography images. Phys Med Biol, 1996;41:495-508.

Gordon CL, Webber CE, Christoforou N, Nahmias C. In-vivo assessment of trabecular bone structure at the distal radius from high-resolution magnetic resonance images. Med Phys, 1997;24(4): in press.

Guglielmi G, Grimston SK, Fischer KC, and Pacifici R. Osteoporosis: Diagnosis with lateral and posterioanterior dual x-ray absorptiometry compared with quantitative CT. Radiology, 1994;192:845-850.

Hahn M, Vogel M, Pompesius-Kema, and Delling G. Trabecular Bone Pattern Factor - a new parameter for simple quantification of bone microstructure. Bone, 1992;13:327-330.

Hangartner TN and Overton TR. Quantitative measurements of bone density using computed tomography. J Comput Assist Tomogr, 1982;6:1156-1162.

Hans D, Schott AM, and Meunier PJ. Ultrasonic assessment of bone: A review. Eur J Med, 1993;2:157-163.

Harma M and Karjalainen P. Trabecular osteopenia in Colles' fracture. Acta Ortho Scand, 1986;57:38-40.

Heaney RP, Avioli LV, Chestnut CH, Lappe J, Recker RR, and Brandenburger GH. Osteoporotic bone fragility detection by ultrasound transmission velocity. JAMA, 1989;261:2986-2990.

Herman GT. Image Reconstruction from projections: The fundamentals of computed tomography. Academic Press, New York, 1980.

Hosie CJ and Smith DA. Precision of measurement of bone density with a special purpose computed tomography scanner. Br J Radiol, 1986;59:345-350.

Hui SL, Slemeda CW, and Johnston CC. Age and bone mass as predictors of fracture in a prospective study. J Clin Invest, 1988;81:1804-1809.

Jara H, Wehrli FW, Chung H, and Ford JC. High-resolution variable flip angle 3D MR imaging of trabecular microstructure in vivo. Magn Res Med, 1993;29:528-539.

Jensen JS and Tondevold E. Mortality after hip fractures. Acta Orthop Scand, 1979;50:161-167.

Jensen KS, Mosekilde L, and Mosekilde L. A model of vertebral trabecular bone architecture and its mechanical properties. Bone, 1990;11:417-423.

Kane JW and Sternheim MM. Physics. (Wiley and Son) ch 20, 1983.

Kiel DP, Felson DT, Anderson JJ, Wilson PWF, and Moskowitz MA. Hip fracture and the use of estrogens in postmenopausal women. New Eng J Med, 1987;317:1169-1174.

Kleerekoper M, Villanueva AR, Stanciu J, Sudhaker D, and Parfitt AM. The role of three-dimensional trabecular microstructure in the pathogenesis of vertebral compression fractures. Calcif Tissue Int, 1985;37:594-597.

Laval-Jeantet AM, Bergot C, Williams M, Davidson K, and Laval-Jeantet M. Dual-energy absorptiometry of the calcaneus: Comparison with vertebral dual-energy x-ray absorptiometry and quantitative computed tomography. Calcif Tiss Int, 1995;56:14-18.

Majumdar S, Weinstein RS, and Prasad RR. Application of fractal geometry techniques to the study of trabecular bone. Med Phys, 1993;20:1611-1619.

Majumdar S, Genant HK, Grampp S, Jergas MD, Newitt DC, and Gies AA. Analysis of trabecular bone structure in the distal radius using high resolution MRI. Eur Radiol, 1994;4:517-524.

Majumdar S and Genant HK. A review of the recent advances in magnetic resonance imaging in the assessment of osteoporosis. Osteo Int, 1995;5:79-92.

Majumdar S, Newitt D, Jergas M, Gies A, Chiu E, Osman D, Keltner J, Keyak J, and H. Genant H. Evaluation of technical factors affecting the quantitation of trabecular bone structure using magnetic resonance imaging. Bone, 1995;17:417-430.

Mallmin H, Ljunghall S, Persson I, Naessen T, Krusemo U, and Bergstrom R. Fracture of the distal forearm as a forecaster of subsequent hip fracture: A population-based cohort study with 24 years of follow-up. Calcif Tissue Int, 1993;52:269-272.

Mazzes RB. On aging bone loss. Clin Orthop, 1981;165:239-252.

Mazzes RB, Barden HS, Eberle RW, and Drue DM. Age changes of spine density in posterior-anterior and lateral projections in normal women. Calcif Tissue Int, 1995;56:201-205.

McCloskey EV, Murray SA, Miller C, Charlesworth D, Tindale W, O'Doherty DP, Bickerstaff DR, Hamdy AT, and Kanis JA. Broadband ultrasound attenuation in the os calcis: Relationship to bone mineral at other skeletal sites. Clinical Science, 1990;78:227-233.

Mosekilde L. Age related changes in vertebral trabecular bone architecture-assessed by a new method. Bone, 1988;9:247-250.

Muller A, Ruegsegger P, and Seitz P. Optimal CT settings for bone evaluations. Phys Med Biol, 1985;30 401-409.

Muller A, Ruegsegger E, and Ruegsegger P. Peripheral QCT: a low-risk procedure to identify women predisposed to osteoporosis. Phys Med Biol, 1989;34:741-749.

Muller A and Ruegsegger P. Analysis of mechanical properties of cancellous bone under conditions of simulated bone atrophy. J Biomechanics, 1996;29:1053-1060.

Mundinger A, Wiesmeier B, Dinkel E, Helwig A, Beck A, and Moenting JS. Quantitative image analysis of vertebral body architecture- improved diagnosis in osteoporosis based on high-resolution computed tomography. Br J Radiol, 1993;66:209-213.

Myers ER, Sebeny EA, Hecker AT, Corcoran TA, Hipp JA, Greenspan SL, and Hayes WC. Correlations between photon absorption properties and failure load of the distal radius in vitro. Calcif Tissue Int, 1991;49:292-297.

Nicholson PH, Haddaway MJ, and Davie MW. The dependence of ultrasonic properties on orientation in human vertebral bone. Phys Med Biol, 1994;39:1013-1024.

Nuti R and Martini G. Measurements of bone mineral density by DXA total body absorptiometry in different skeletal sites in postmenopausal osteoporosis. Bone, 1992;13:173-178.

Odgaard A and Gundersen HJG. Quantification of connectivity in cancellous bone, with special emphasis on 3-D reconstructions. Bone, 1993;14:173-182.

Orwoll ES and Oviatt SK. Longitudinal precision of dual-energy x-ray absorptiometry in a multicenter study. J Bone Miner Res, 1991;6:191-197.

Ott SM. When bone mass fails to predict bone failure. Calcif Tissue Int, 1993;53(S1):S7-S13.

Overgaard K, Hansen MA, Riis BJ, and Christiansen C. Discriminatory ability of bone mass measurements (SPA and DEXA) for fractures in elderly postmenopausal women. Calcif Tissue Int, 1992;50:30-35.

Owen RA, Melton LJ, Johnson KA, Ilstrup DM, and Riggs BL. Incidence of Colles' fracture in a North American community. Am J Public Health, 1982;72:605-607.

Parfitt AM, Mathews CH, Villanueva AR, and Kleerekoper M. Relationship between surface, volume, and thickness of iliac trabecular bone in aging and in osteoporosis. J Clin Invest, 1983;72:1396-1409.

Parisien M, Mellish RW, Silverberg SJ, Shane E, Lindsay R, Bilezikian JP, and Dempster DW. Maintenance of cancellous bone connectivity in primary hyperparathyroidism: trabecular strut analysis. J Bone Miner Res, 1992;7:913-919.

Pouilles JM, Tremollieres F, Todorovsky N, and Ribot C. Precision and sensitivity of dual-energy x-ray absorptiometry in spinal osteoporosis. J bone Miner Res,1992;6:997-1002.

Reinbold WD, Genant HK, Reiser U, Harris ST, and Ettinger B. Bone mineral content in early-postmenopausal and postmenopausal osteoportic women: Comparison of measurement methods. Radiology, 1986;160:469-478.

Riggs BL, Hodgson SF, O'Fallon WM, Chao EY, Wahner HW, Muhs JM, Cedel SL, and Melton ML. Effect of fluoride treatment on the fracture rate in postmenopausal women with osteoporosis. N Engl J Med, 1990;322:802-809.

Robinson GS. Edge detection by compass gradient masks. Comput Graphics Image Processing, 1977;6:492-501.

Rockoff D, Scandrett J, and Zacher R. Quantitation of relevant image information: automated radiographic bone trabecular Characterization. Radiology, 1971;101:435-439.

Ruegsegger P, Koller B, and Muller R. A microtomographic system for the nondestructive evaluation of bone architecture. Calcif Tissue Int, 1996;58:24-29.

Sahoo PK, Soltani S, and Wong AK. A survey of thresholding techniques. Comput Vision Graphics Image Process, 1988;41:233-260.

Schild HH. MRI made easy. Edited by Berlex Canada INC. Berlin: Nationales Press; 1990.

Schlenker RA and VonSeggen WW. The distribution of cortical and trabecular bone mass along the lengths of the radius and ulna and its implications for in-vivo bone mass measurements. Calcif Tissue Res, 1976;20:41-52.

Schenck JF, Hart HR, Foster TH, and Edelstein WA. Improved MR imaging of the orbit at 1.5 T with surface coils. Am J Roentgenol, 1985;144:1033-1036.

Sebag GH and Moore SG. Effect of trabecular bone on the appearance of marrow in gradient echo imaging of the appendicular skeleton. Radiology, 1990;174:855-859.

Shepp LA and Logan BF. The Fourier reconstruction of a head section. IEEE Trans Nucl Sci, 1974;21:21-43.

Solgaard S and Petersen VS. Epidemiology of distal radius fractures. Acta Orthop Scand, 1985;56:391-393.

Sparado JA, Werner FW, Brenner RA, Fortino MD, Fay LA, and Edwards WT. Cortical and trabecular bone contribution to the osteopenic distal radius. J Orthop Res, 1994;12:211-218.

Stebler B and Ruegsegger P. Special purpose CT-system for quantitative bone evaluation in the appendicular skeleton. Biomed. Technik, 1983;28:196-265.

Takagi Y, Fujii Y, Miyauchi A, Goto B, Takahashi K, and Fujita T. Transmenopausal change of trabecular bone density and structural pattern assessed by peripheral quantitative computed tomography in Japanese women. J Bone Min res, 1995;10:1830-1834.

Thomas TG and Steven RS. Social effects of fractures of the neck of the femur. Br Med J, 1974;3:456-458.

Townsend PR, Raux P, Rose RM, Miegel RE, and Radin EL. The distribution and anisotropy of the stiffness of cancellous bone in the human patella. J Biomechanics, 1975;8:363-367.

Verly JG and Bracewell RN. Blurring in tomograms made with X-ray beams of finite width. J Comput Assist Tomogr, 1979;3:662-678.

Vesterby A, Gundersen HJG, Melsen F, and Mosekilde L. Normal postmenopausal women show iliac crest trabecular thinning on vertical sections. Bone, 1989a;10:333-339.

Vesterby A, Gundersen HJG, Melsen F, and Mosekilde L. Star volume of marrow space and trabecular of the first lumbar vertebra: Sampling efficiency and biological variation. Bone, 1989b;10:7-13.

Wasserman SH and Burzel US. Osteoporosis: The state of the art in 1987: A review. Sem Nucl Med, 1987;17:282-292.

Webber CE. The effect of fat on bone mineral measurements in normal subjects with recommended values of bone, muscle, and fat attenuation coefficients. Clin Phys Physiol Meas, 1987;8:143-158.

Wehrli FW, Ford JC, Attie M, Kressel HY, and Kaplan FS. Trabecular structure: preliminary application of MR interferometry. Radiology, 1991;171:615-621.

Wehrli FW, Ford JC, Chung HW, Wehrli SL, Williams JL, Grimm MJ, Kugelmass SD, and Jara H. Potential role of nuclear magnetic resonance for the evaluation of trabecular bone quality. Calcif Tissue Int, 1993;53:S162-S169.

Wehrli FW, Ford JC, and Haddad JG. Osteoporosis: Clinical assessment with quantitative MR imaging in diagnosis. Radiology, 1995;196:631-641.

Weiss NS, Ure CL, Ballard JH, Williams AR, and Daling JR. Decreased risk of fractures of the hip and lower forearm with postmenopausal use of estrogen. New Eng J Med, 1980;303:1195-1198.

Whitehouse WJ. Cancellous bone in the anterior part of the iliac crest. Calcif Tissue Res, 1977;23:67-76.

Wong EC, Jesmanowicz AJ, and Hyde JS. High-resolution, short echo time MR imaging of the fingers and wrist with a local gradient coil. Radiology, 1991;181:393-397.

Wood ML, Bronskill MJ, Mulkern RV, and Santyr GE. Physical MR desktop data. J Magn Reson Imaging, 1993;3:S19-S48.

Wu Z, Chung HW, and Wehrli FW. A bayesian approach to subvoxel tissue classification in NMR microscopic images of trabecular bone. Magn Reson Med, 1994;31:302-308.

Yester MV and Barnes GT. Geometrical limitations of computed tomography (CT) scanner resolution. Proc SPIE, Appl Opt Instr in Medicine, 1977;127:296-303.

Yidong C, Dougherty ER, Totterman SM, and Hornak JP. Classification of trabecular structure in magnetic resonance images based on morphological granulometries. Magn Res Med, 1993;29:358-370.

Zhang TY and Suen CY. A fast parallel algorithm for thinning digital patterns. Commun Assoc Comput Mach, 1984;27:236-239.

APPENDIX A

JOURNAL ARTICLES (Peer Reviewed):

1. **Gordon CL**, Webber CE, Adachi JD, and Christoforou N. In vivo assessment of trabecular bone structure at the distal radius from high-resolution computed tomography images. Phys Med Biol, 1996;41:495-508.

2. **Gordon CL**, Webber CE, Christoforou N, and Nahmias C. In vivo assessment of trabecular bone structure at the distal radius from high-resolution magnetic resonance images. Med Phys, 1997;24(4): (in press).

JOURNAL ARTICLES (Submitted):

Gordon CL and Webber CE. In vitro assessment of trabecular bone structure: Correlation with stress testing. Submitted to Magnetic Resonance Analysis (March, 1997).

ABSTRACTS AND PRESENTATIONS:

1. **Gordon CL** and Webber CE. High-resolution MR and Peripheral CT Imaging Applied to the Study of Trabecular Bone Structure. Presented at the 6th Annual Radiology Symposium, Chedoke-McMaster Hospitals, Department of Radiology, Hamilton, ON, March 1994 (awarded first prize in graduate student competition).

2. **Gordon CL**, Adachi JD, and Webber CE. Assessment of Trabecular Bone Structure.
   (a) Presented at the American Society for Bone and Mineral Research, Kansas City, USA, September 1994.
   (b) Abstracted in J Bone Miner Res, 1994;9(S1) S404.

3. **Gordon CL**, Adachi JD, and Webber CE. Assessment of Trabecular Bone Structure by pQCT.
   (a) Presented at the Annual Meeting of the Canadian Organization of Medical Physicists, Montreal, Canada, June 1995.
   (b) Abstracted in Medical Physics, 1995;5 p678.

4. **Gordon CL**, Adachi JD, and Webber CE. In Vivo Assessment of Trabecular Bone Structure.
   (a) Presented at the American Society for Bone and Mineral Research, Baltimore, USA, September 1995.
   (b) Abstracted in J Bone Miner Res, 1995;**10**(S1) S472.

5. **Gordon CL**. Derivation of Indices of Trabecular Bone Structure from Peripheral Quantitative Computed Tomography Images.
   Invited presentation to the Peripheral Quantitative Computed Tomography 2nd User's Meeting, Tampa Bay, USA, Feburary 1996.

6. **Gordon CL**. Beyond Bone Mineral Mass: The Use of X-ray CT and MR Imaging in the Study of Bone Morphology.
   Invited presentation to the Radiology Resident Rounds, Chedoke-McMaster Hospitals, Department of Radiology, Hamilton, ON, March 1996.

7. **Gordon CL**, MacIntyre NJ, and Webber CE. Bone Mass and Structure Deficits in the Non-fractured Radius of Patients with Colles Fracture.
   Presented at the Fifth Bath Conference on Osteoporosis and Bone Mineral Measurement, Bath, UK, June 1996.

8. **Gordon CL** and Webber CE. Assessment of Trabecular Bone Structure in the Radius by MRI and CT: Correlation with Stress Testing.
   (a) Presented at the Annual Meeting of the American Association of Physicists in Medicine, Philadelphia, USA, July 1996.
   (b) Abstracted in Medical Physics, 1996;**6** p1087.

9. **Gordon CL**, MacIntyre NJ, and Webber CE. In Vivo Assessment of Trabecular Bone Structure.
   (a) Presented at the annual meeting of The American Society for Bone and Mineral Research, Seattle, USA, September 1996.
   (b) Abstracted in J Bone Miner Res, 1996;**11**(S1) S473.

APPENDIX B

C CODE FOR pQCT TO MUMC DISPLAY FILE CONVERSION

This routine converts the raw pQCT image data file which can be downloaded from the scanner onto a diskette. The data file is then copied onto a Sun workstation which runs this conversion program "wrist2". Wrist2 converts the raw pQCT data file into a data format which is readable by MUMC display.

```
#include <stdio.h>
#include <fcntl.h>
#include <malloc.h>
main(argc, argv)
int argc;
char *argv[];

{  int i,j;
   short int *buf;
   int fd1, fd2;
    short  int image[256][256];
    short int image2[256][256];
    short int image3[256][256];
        unsigned char *ps,*pt;

   if (argc<2) {  printf("TYPE: rdimg <file1> \n \n"); exit(0); }
   printf("\n");

   fd1=open(argv[1], 0);
if (fd1==-1) { printf("\n error \n"); exit(0); }

   fd2=open(argv[2],O_RDWR|O_CREAT,0000644);

   for (i=0; i<254;i++)
        {
          read(fd1,   ( char*)image[i],508);

        }
/* Swap bytes for DOS to UNIX transfer */
        ps=image[0];
        pt=image2[0];
       swab(ps,pt,2*65536);

/* Threshold to get rid of negative CT values */
        for (i=0;i<256;i++)
        for(j=0;j<256;j++)
        { if (((int) image2[i][j] > 2500 )| ((int)image2[i][j] <0))
                image2[i][j]=0;}

/* Swap rows and columns */
        for (i=0;i<256;i++)
        for(j=0;j<256;j++)
         {image3[j][i] =image2[i][j];}
printf("Writing to MUMC DISPLAY ....\n");
   for (i=255; i>=0;i--)
        { write(fd2, (char*)image3[i],512);}


   close(fd1);
   close(fd2);
   exit(0);

}
```

APPENDIX C

C CODE FOR pQCT IMAGE SEGMENTATION

The segmentation algorithm is named "stratec3". The user is prompted with a menu of segmentation options during analysis.

```
#include <stdio.h>
#include <fcntl.h>
#include <math.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/uio.h>

unsigned char struct_element[2][2]={
                                        1,1,
                                        1,1

                                    };
main(argc, argv)
int argc;
char *argv[];
  { int i,j;
    FILE *results;
        char *buf2,fpoint2;
        char out_file[25],in_file[25];
        char subject_name1[25],subject_name2[25],output_file[25];
        char comment1[75],comment2[75],comment3[75];
    short int *buf;
    int fd1, fd2;
    short  int image[256][256],out_image[256][256],out_image2[256][256];
    short in_image[256][256];
    short int temp_image[256][256],in_image2[256][256], image_temp[256][256];
    int msk[9],mm[5][5],holes[500],hole_number,k,ends;
    short corners[10],x1,x2,y1,y2;
    short int x_c,y_c;
    short display_size,total_area,size=256,c,nd,std_ct;
    int mean_soft,soft,sum_soft_t,soft_t,cortical_t,out_of_plane;
    int bone_contour=0,trab_area=0,cort_area=0,outer_contour=0,cortical_try=0,contour_flag;
        int contour_breaks,y_1,y_2,x_1,x_2;
  short sum_std,cortical;
  char *in,*out;
  double trab_t;
  float mean,std,trabecular_area,cortical_area,total_bone_area,bone_pixels,TBV,PERI_C,ENDO
        float CRT_THK, R_out, R_in,PERI_CIRC,ENDO_CIRC, I_CIRC;
  char reply;
   int x=1,y=1;
        if (argc<2) {  printf("TYPE: stratec2 <file1> \n \n"); exit(0); }
        printf("\n");

        fd1=open(argv[1], 0);
        if (fd1==-1) { printf("\n error \n"); exit(0); }

         fd2=open(argv[2], 1);

        for (i=0; i<256;i++){
                read(fd1,   ( char*)image[i],512);
        }

        for (i=0;i<size;i++)
                for(j=0;j<size;j++) {
                in_image[i][j]=image[i][j];
                /* printf("%d", image[i][j]); */
        }

/* open output file results.dat */
        printf("Enter name of output file to write results:> ");
        scanf("%24s",output_file);
        printf("Name of subject being analyzed:   \n");
        scanf("%s %s",subject_name1,subject_name2); getchar();
        printf("Input additional comments (3 lines expected):   \n");
        gets(comment1);
        gets(comment2);
```

```
        gets(comment3);
        if ((results=fopen(output_file,"w")) == (FILE *) NULL){
                printf("File Error");
                exit(1);
        }
        fprintf(results,"RESULTS  For:  %s  %s\n",subject_name1,subject_name2);
        fprintf(results,"         File: %s\n",output_file);
        printf(results," \n");
        fprintf(results,"----------------COMMENTS-------------------------\n");
        fputs(comment1,results);
        fprintf(results,"\n");
        fputs(comment2,results);
        fprintf(results,"\n");
        fputs(comment3,results);
        fprintf(results,"\n");
        fprintf(results,"-----------------------------------------------\n");
do {
                printf("\n <---- BONE STRUCTURE ANALYSIS ----> \n");
                printf("\n          ( MAIN MENU ) \n\n");
                printf("        a. - Define ROI sub-image ..... (#10)\n");
                printf("        b. - Region Grow ............... (#11)\n");
                printf("        c. - Contour Bone .............. (#12)\n");
                printf("        d. - Calculate Trabecular area .. (#13)\n");
                printf("        e. - Moments of Inertia ........ (#14)\n");
                printf("        f. - Adaptive threshold ........ (#15)\n");
                printf("        g. - Count holes in network \n");
                printf("        h. - Skeleton by ZHANG-SUEN ..... (#17)\n");
                printf("        i. - Indices of structure ..... (#20)\n");
                printf("        q. - Quit\n");
                printf("\n What is your selection ? ");
                reply=getchar();
                printf("\n");

                switch(reply) {
                        case 'a':
                                region_of_interest(image,out_image,corners);
                                x1=corners[0];  y1=corners[1];
                                x2=corners[2]; y2=corners[3];
                                out="/MUMC/remote1/555_10.img";
                                write_image(out,out_image,256);
                                break;
                        case 'b':
                                x=(x1+x2)/2; y=(y1+y2)/2;
                                printf("\n Growing region....");
                                maximum(out_image,size,corners);
                                x=corners[0]; y=corners[1];
                                cortical_t=corners[2];
                                cortical_t=0.45*cortical_t;
                                soft_t=1.25*15*histogram(out_image,256,15);
                                soft_t=0.5*cortical_t;
                                soft_t=400;   /* soft tissue threshold */
                                total_area=grow_region(out_image,256,256,x,y,soft_t);

/* Make sure that a big enough area is grown */
                                c=1;
                                while (total_area <500){
                                        in="/MUMC/remote1/555_10.img";
                                        read_image(in,out_image,256);
                                        total_area=grow_region(out_image,256,256,y+c,x+c,so
                                        c++;
                                }
                                for (i=y1;i<=y2;i++)
                                   for (j=x1;j<=x2;j++) {
                                        if(out_image[i][j]==3000)
                                        else {
                                                out_image[i][j]=0;
```

```
                                        sum_soft_t=sum_soft_t+in_image[i][j];
                                        }
                                    }
;
                                        mean_soft=(sum_soft_t/soft);
                                    out="/MUMC/remote1/555_11.img";
                                    write_image(out,out_image,256);
                                    break;
                            case 'c':

                                    in="/MUMC/remote1/555_10.img";
                                    read_image(in,out_image,256);
                                    clean_image(out_image,256);
 /* calculate cortical threshold */
                                    maximum(out_image,size,corners);
                                    x=corners[0]; y=corners[1];
                                    cortical_t=corners[2];
                                    cortical_t=0.4*cortical_t;
                                    /*cortical_t=600;*/
                                    cortical_t=cortical_t-(cortical_try*20);
                                    printf(" Cortical Threshold=%d\n",cortical_t);
                                    for (i=1; i<=9; i++)
                                            msk[i]=1;
                                    convolve_mask(out_image,msk,9,x1,x2,y1,y2);
                                    for (i=0;i<size;i++)
                                        for (j=0;j<size;j++) {
                                             temp_image[i][j]=out_image[i][j];
                                             out_image[i][j]=0;
                                         }
                                    /*printf(" Input cortical and subcortical T\n");
                                    scanf("%d",&cortical_t);*/
                                    get_cortex(temp_image,out_image,size,x1,x2,y1,y2,cortical_t
 /* remove isolated bits of bone */
                                    clean_image(temp_image,256);

                                    grow_region(temp_image,256,256,x,y,100);
                                    for (i=y1;i<=y2;i++)
                                       for (j=x1;j<=x2;j++) {
                                            if(temp_image[i][j]==3000)
                                            else {
                                                    temp_image[i][j]=0;
                                            }
                                       }*/
                                    out="/MUMC/remote1/555_12.img";
                                    write_image(out,temp_image,256);
                                    printf("  \n");
                                    printf("          ****************************************
                                    printf("          * OPEN DISPLAY AND VERIFY THAT CONTOUR IS
                                    printf("          *                    IN IMAGE #12
                                    printf("          ****************************************
                                    printf("  \n");
                                    printf("Is contour broken  ? (1=YES  0=NO)\n");
                                    scanf("%d",&contour_flag);
                                    if(contour_flag==1){
                                            printf("How many sets of breaks to close ?\n");
                                            scanf("%d",&contour_breaks);
                                            for (k=1; k<= contour_breaks; k++){
                                                    printf("Input co-ordinates of end #1\n");
                                                    scanf("%d %d",&y_1,&x_1);   /* ordinates ar
                                                    printf("Input co-ordinates of end #2\n");
                                                    scanf("%d %d",&y_2,&x_2);
                                                    printf("Closing ..... break # %d\n",k);
                                                    close_contour(temp_image,size,x_1,x_2,y_1,y
                                            }

                                    }
                                    out="/MUMC/remote1/555_12.img";
```

```
                                write_image(out,temp_image,256);
/* remove isolated bits of bone */
                                grow_region(temp_image,256,256,x,y,100);
                                for (i=y1;i<=y2;i++)
                                   for (j=x1;j<=x2;j++) {
                                        if(temp_image[i][j]==3000)
                                        else {
                                                temp_image[i][j]=0;
                                        }
                                   }
                                out="/MUMC/remote1/555_12.img";
                                write_image(out,temp_image,256);
/* outline inner and outer contour */
                                in="/MUMC/remote1/555_12.img";
                                read_image(in,out_image,256);
                                for (i=0;i<size;i++)
                                   for (j=0;j<size;j++) {
                                        if (out_image[i][j]==255){
/* count cortical area */                                       cort_area++;
                                        }
                                         temp_image[i][j]=0;
                                   }
                                distance_transform(out_image,256);
                                for (i=0;i<size;i++)
                                   for (j=0;j<size;j++) {
                                        if(out_image[i][j]==2) {
                                                temp_image[i][j]=255;
                                                bone_contour++;
                                        }
                                   }
                                out="/MUMC/remote1/555_14.img";
                                write_image(out,temp_image,256);
/* define outer contour */
                                in="/MUMC/remote1/555_12.img";
                                read_image(in,temp_image,256);
                                for (i=0;i<size;i++)
                                        for (j=0;j<size;j++) {
                                                out_image[i][j]=0;
                                        }
                                contour(temp_image,out_image,size,x1,x2,y1,y2,cortical_t);
                                out="/MUMC/remote1/555_22.img";
                                write_image(out,out_image,256);
                                for (i=0;i<size;i++)
                                   for (j=0;j<size;j++) {
                                        if(out_image[i][j]==255) {
                                                outer_contour++;
                                        }
                                   }

                                /*printf("Cortical and subcortical bone area= %d\n",cort_ar
                                break;

                        case 'd':
                                in="/MUMC/remote1/555_12.img";
                                read_image(in,out_image,256);
                                x=(x1+x2)/2; y=(y1+y2)/2;
                                /* printf("\nEntering x=%d  y=%d", y,x); */
                                total_area=fill_contour(out_image,256,256,y,x,254);
                                /*printf("totalcross-sectional area=  %d\n", total_area);*/
                                in="/MUMC/remote1/555_11.img";
                                read_image(in,in_image2,256);
                                in="/MUMC/remote1/555_10.img";
                                read_image(in,image,256);
                                for (i=0;i<size;i++)
                                        for(j=0;j<size;j++) {
                                                if(out_image[i][j]==3000){
```

```
                                        image_temp[i][j]=image[i][j];
                                        if (in_image2[i][j] >0){
                                                trab_area++;
;                                       }
                                        else {
                                                out_image[i][j]=0;
                                                image_temp[i][j]=0;
                                        }
/* locate all bone pixels that are not 8-connected */
                                        if((out_image[i][j]==0)&&(image_temp[i][j]> soft_t)
                                                out_image2[i][j]=255;
                                                out_of_plane=out_of_plane+1;
                                        }
                                        else {
                                                out_image2[i][j]=0;
                                        }

                                }
                                printf("Out-plane pixels= %d\n",out_of_plane);
                                fprintf(results,"Number of Out-plane pixels= %d\n",out_of_p
                                bone_pixels=((float) trab_area+(float)out_of_plane)*0.33*0.
                                total_area=total_area;   /* total cross-sectional area */
/* remove inner contour from trabecular area count */
                                trabecular_area=(float)total_area-((float)bone_contour-(flc
                                cortical_area=0.33*0.33*(float)cort_area;
                                trabecular_area=trabecular_area*0.33*0.33;
                                total_bone_area=cortical_area+trabecular_area;
/* Calculate circular ring parameters */
                                R_out=sqrt(total_bone_area/3.14159);
                                R_in=sqrt(trabecular_area/3.14159);
                                CRT_THK=R_out-R_in;
                                PERI_CIRC=2*3.14159*R_out;
                                ENDO_CIRC=2*3.14159*R_in;
                                I_CIRC=0.785398*((R_out*R_out*R_out*R_out)-(R_in*R_in*R_in*
                                        printf("Total bone area= %.2f mm2\n", total_bone_ar
                                        printf("Cortical(subcort) area= %.2f mm2\n", cortic
                                        printf("Trabecular area= %.2f mm2\n", trabecular_ar
                                        fprintf(results,"Total bone area= %.2f mm2\n", tota
                                        fprintf(results,"Cortical(subcort) area= %.2f mm2\n
                                        fprintf(results,"\nTrabecular area= %.2f mm2\n", tr
                                TBV=(bone_pixels/trabecular_area)*100.0;
                                        printf("Trabecular Bone Volume (TBV)= %.2f %%\n", T
                                        fprintf(results,"Trabecular Bone Volume (TBV)= %.2f
                                        fprintf(results,"\n");
                                PERI_C=0.33*(float)outer_contour;
                                ENDO_C=0.33*((float)bone_contour-(float)outer_contour);
                                        printf("Periostal circumference= %.2f mm\n",PERI_C)
                                        printf("Endostal circumference= %.2f mm\n",ENDO_C);
                                fprintf(results,"----------------GEOMETRICAL PARAMETERS----
                                fprintf(results,"\n");
                                fprintf(results," 1. CIRCULAR RING MODEL  \n");
                                fprintf(results,"       Cortical Thickness= %.2f mm\n",CRT_
                                fprintf(results,"       Periostal Circumference= %.2f mm\n"
                                fprintf(results,"       Endostal Circumference= %.2f mm\n",
                                fprintf(results,"       Axial Moment of Inertia= %.3f mm4\n
                                fprintf(results,"\n");
                                fprintf(results," 2. REAL SHAPE  \n");
                                fprintf(results,"       Periostal Circumference= %.2f mm\n"
                                fprintf(results,"       Endostal Circumference= %.2f mm\n",
                                out="/MUMC/remote1/555_13.img";
                                write_image(out,out_image,256);
                                out="/MUMC/remote1/555_16.img";
                                write_image(out,out_image2,256);
                                break;

                        case 'e':
```

```
                    in="/MUMC/remote1/555_12.img"; /* read in cortical and sub
                    read_image(in,in_image2,256);
                    in="/MUMC/remote1/555_10.img"; /* get original ct values *
                    read_image(in,in_image,256);

                    for (i=0;i<size;i++)
                        for (j=0;j<size;j++) {
                            out_image[i][j]=0;
                            if(in_image2[i][j]==255){
                                out_image[i][j]=in_image[i][j];
                            }
                        }
                    x=(x1+x2)/2; y=(y1+y2)/2;
                    total_area=fill_contour(in_image2,256,256,y,x,254);
                    for (i=0;i<size;i++)
                        for(j=0;j<size;j++) {
                            if(in_image2[i][j]==3000){
                            }
                        }
                    out="/MUMC/remote1/555_21.img";
                    write_image(out,out_image,256);
                    moment_of_inertia(out_image,out_image,size,x1,x2,y1,y2,mm,c
                    /*x_c=mm[1][0]/mm[0][0]; y_c=mm[0][1]/mm[0][0];
                    printf("Centroid of cortical shell = %d  %d\n",x_c,y_c);*/
                    break;

                case 'f':
                    /* read in filled contour image */
                    in="/MUMC/remote1/555_13.img";
                    read_image(in,in_image2,256);
                    in="/MUMC/remote1/555_13.img";
                    read_image(in,out_image,256);
                    optimize_contrast(out_image,256,x1,x2,y1,y2,corners[2]);
                      msk[1]=0; msk[2]=-1; msk[3]=0;
                      msk[4]=-1; msk[5]=5; msk[6]=-1;
                      msk[7]=0; msk[8]=-1; msk[9]=0;
                    convolve_mask(out_image,msk,1,x1,x2,y1,y2);
                    thresholding(out_image,in_image2,256,cortical_t);
                    out="/MUMC/remote1/555_15.img";
                    write_image(out,out_image,256);
/* Define only inner contour */
                    in="/MUMC/remote1/555_22.img";
                    read_image(in,in_image2,256);
                    in="/MUMC/remote1/555_14.img";
                    read_image(in,out_image,256);
                    subtract_images(out_image,in_image2,size);
                    out="/MUMC/remote1/555_8.img";
                    write_image(out,out_image,256);
/* add inner contour to trabecular bone */
                    in="/MUMC/remote1/555_15.img";
                    read_image(in,in_image2,256);
                    in="/MUMC/remote1/555_8.img";
                    read_image(in,out_image,256);
                    add_images(out_image,in_image2,size);
                    out="/MUMC/remote1/555_15.img";
                    write_image(out,out_image,256);
                    break;

                case 'g':
                    in="/MUMC/remote1/555_15.img";
                    read_image(in,out_image,256);
                    in="/MUMC/remote1/555_16.img";
                    read_image(in,temp_image,256);
                    add_images(temp_image,out_image,256);
                    out="/MUMC/remote1/555_19.img";
                    write_image(out,temp_image,256);
```

```
                                in="/MUMC/remote1/555_19.img";
                                read_image(in,out_image,256);
/* label border of image with value=200 */
                                for (i=y1;i<=y2;i++)
                                  for (j=x1;j<=x2;j++) {
                                        temp_image[i][j]=out_image[i][j];
                                                temp_image[y1][j]=200;
                                                temp_image[y2][j]=200;
                                                temp_image[i][x1]=200;
                                                temp_image[i][x2]=200;
                                  }
fprintf(results,"----------------INDICES OF STRUCTURE-------------------------\n");
                hole_number=hole_counter(holes,temp_image,x1,x2,y1,y2)-1;
                printf(" holes detected= %d\n",hole_number);
                fprintf(results,"Number of holes detected= %d\n",hole_number);
                                mean_std(holes,mean,std,hole_number,results);
                                sort(holes,hole_number,results);
                                break;
                        case 'h':
/* skeleton in plane-connected bone structure  by Zhang-Suen algorithm */
                                in="/MUMC/remote1/555_15.img";
                                read_image(in,out_image,256);

                                thinzs(out_image,255);
                                out="/MUMC/remote1/555_17.img";
                                write_image(out,out_image,256);

/* skeleton out of plane bone structure  by Zhang-Suen algorithm */
                                in="/MUMC/remote1/555_16.img";
                                read_image(in,temp_image,256);

                                thinzs(temp_image,255);
                                out="/MUMC/remote1/555_18.img";
                                write_image(out,temp_image,256);
/* add two skeletons together */
                                add_images(out_image,temp_image,size);
                                out="/MUMC/remote1/555_17.img";     /* overwrite image */
                                write_image(out,out_image,256);
                                break;


                        case 'i':
                                /* Calculate indices for in-plane structure */
                                in="/MUMC/remote1/555_17.img";
                                read_image(in,out_image,256);
                                for (i=0;i<size;i++)
                                   for (j=0;j<size;j++) {
                                        temp_image[i][j]=out_image[i][j];
                                         out_image[i][j]=0;
                                   }
                                printf(" *** IN PLANE STRUCTURE ***\n");
                                indices(temp_image,out_image,256,256,255,300,275,results);
                                out="/MUMC/remote1/555_20.img";
                                write_image(out,out_image,256);

                                /* Calculate indices for out of plane structure
                                in="/MUMC/remote1/555_18.img";
                                read_image(in,out_image,256);
                                for (i=0;i<size;i++)
                                for (j=0;j<size;j++)
                                        {temp_image[i][j]=out_image[i][j];
                                         out_image[i][j]=0;}
                                printf(" *** OUT OF PLANE STRUCTURE ***");
                                indices(temp_image,out_image,256,256,255,300,275);*/
                                break;
                        }
```

```
                } while (reply != 'q');

  /*  Write to MUMC display file */
     for (i=0; i<256;i++)

          {
             write(fd2, (char*)out_image[i],512);
           }

     close(fd1);
     close(fd2);
     exit(0);
  }

  /********** Function to read a MUMC display image *************/

  int read_image(in_file,values,m_size)
  char *in_file;
  short m_size.values[256][256];
  {
  short int fp,i;
  fp=open(in_file,0);
  for (i=0; i<m_size;i++)
          {
          read(fp, (char*)values[i],m_size*2);
          }
   close(fp);
   }


  /********** Function to write image to MUMC display *************/

  int write_image(out_file,values,m_size)
  char *out_file;
  short values[256][256],m_size;
  {
  int fp,i;
  fp=open(out_file,1);
  for (i=0; i<m_size;i++)
          {
          write(fp, (char*)values[i],m_size*2);
          }

     close(fp);
  }

  /*************** Function to close bone contour ****************/
  int close_contour(values,m_size,rx1,rx2,ry1,ry2)
  short int values[256][256],m_size,rx1,rx2,ry1,ry2;
  {
          int i,j,n=1,f;
          int delta,run_length,threshold,x[3],y[3];
          double slope,f_dec;


          x[1]=rx1; x[2]=rx2;
          y[1]=ry1; y[2]=ry2;

  /* define line between the end points */
          slope= ((double) (y[2]-y[1]))/((double)(x[2]-x[1]));
          printf("slope = %f\n",slope);
  /* draw line between ends */
          for (i=x[1]; i<=x[2]; i++) {
                  for (f=0; f<10; f++) {
                          f_dec=0.1*f;
                          j=slope*(i+f_dec-x[1])+y[1];
```

```
                        values[i][j]=255;
  /* make this missing bit of contour 3 pixel widths thick */
                        values[i][j-1]=255;
                        values[i][j+1]=255;
                        values[i-1][j]=255;
                        values[i+1][j]=255;
                }
        }
  }

  /********** Function to perform a five point smooth ***************/
  int smooth_5(values,m,sf,rx1,rx2,ry1,ry2)
   short int values[256][256],rx1,rx2,ry1,ry2,sf;
  int m[9];
  {  int i,j,k;

      short int in_values[256][256];
      short int values_x[256][256];
          int x=1,h=1,y=1;
          int m1,m2,m3,m4,m5,m6,m7,m8,m9;
          int sum1,sum2,sum3,sum4,sum5,sum6,sum7;
          double sum_mask;

     for (i=ry1;i<= ry2;i++)
          for(j=rx1;j<= rx2;j++) {
                  in_values[i][j]=values[i][j];
          }

          x=ry1+1;
          do {
          y=rx1+1;
          do {
 sum1=(m[1]*values [x-1][y-1])+(m[2]*values[x-1][y])+(m[3]*values[x-1][y+1]);     sum2=(m[4]*
 sum3=(m[7]*values[x+1][y-1])+(m[8]*values[x+1][y])+(m[9]*values[x+1][y+1]);

 sum4=values[x+2][y-2]+values[x+2][y-1]+values[x+2][y]+values[x+2][y+1]+values[x+2][y+2];

 sum5=values[x-2][y-2]+values[x-2][y-1]+values[x-2][y]+values[x-2][y+1]+values[x-2][y+2];

 sum6=values[x+1][y-2]+values[x][y-2]+values[x-1][y-2];
 sum7=values[x+1][y+2]+values[x][y+2]+values[x-1][y+2];
          values_x[x][y]=(sum1+sum2+sum3+sum4+sum5+sum6+sum7)/25;
         if (values_x[x][y] > 32000 )
                  values_x[x][y]=0;
          if (values_x[x][y] < 0) {
                  values_x[x][y]=0;
          }
          y=y+1;
          }
          while (y <=(rx2-1));
          x=x+1;
          }
          while (x<=(ry2-1));

          for (i=ry1; i<=ry2; i++)
          for (j=rx1; j<=rx2; j++) {
                  values[i][j]=values_x[i][j];
                  if (values[i][j] <0)
                          values[i][j]=0;
                  }
}



/********** Function to convolve image with a mask ***************/
int convolve_mask(values,m,sf,rx1,rx2,ry1,ry2)
```

```
   short int values[256][256],rx1,rx2,ry1,ry2,sf;
  int m[9];
  {  int i,j,k;

  ;
      short int in_values[256][256];
      short int values_x[256][256];
          int x=1,h=1,y=1;
          int m1,m2,m3,m4,m5,m6,m7,m8,m9;
          int sum1,sum2,sum3;
          double sum_mask;

     for (i=ry1;i<= ry2;i++)
          for(j=rx1;j<= rx2;j++)
          {
           in_values[i][j]=values[i][j];
          }

          x=ry1+1;
          do
          {
          y=rx1+1;
          do
          {
 sum1=(m[1]*values [x-1][y-1])+(m[2]*values[x-1][y])+(m[3]*values[x-1][y+1]);     sum2=(m[4]*
 sum3=(m[7]*values[x+1][y-1])+(m[8]*values[x+1][y])+(m[9]*values[x+1][y+1]);
          values_x[x][y]=(sum1+sum2+sum3)/sf;
        if (values_x[x][y] > 32000 )
                  values_x[x][y]=0;
         if (values_x[x][y] < 0)
         {
                  values_x[x][y]=0;
         }
         y=y+1;
         }
         while (y <=(rx2-1));
         x=x+1;
         }
         while (x<=(ry2-1));

         for (i=ry1; i<=ry2; i++)
         for (j=rx1; j<=rx2; j++)
                 {
                 values[i][j]=values_x[i][j];
                 if (values[i][j] <0)
                         values[i][j]=0;
                 }
}


/********** Function to apply median filter to image *****************/
int median_filter(values,in_values,m_size)
short int values[256][256],m_size,in_values[256][256];
{  int i,j,k;
    short int out_values[256][256],m[9];
        int x=1,h=1,y=1,lp,size;
        int row, column, num_median;
        int temp;

        row=m_size;
        column=m_size;
/* *** Apply median filter  *** */
        x=2;
        do
        {
        y=2;
        do
```

```
        {
         m[0]=values [x-1][y-1];
         m[1]=values[x-1][y];
         m[2]=values[x-1][y+1];
         m[3]=values[x][y-1];
         m[4]=values[x][y];
         m[5]=values[x][y+1];
         m[6]=values[x+1][y-1];
         m[7]=values[x+1][y];
         m[8]=values[x+1][y+1];
                for (i=0; i < 8; ++i)
                {
                        for(j=i+1; j < 9; ++j)
                        {
                                if (m[i] > m[j])
                                {
                                        temp=m[i];
                                        m[i]=m[j];
                                        m[j]=temp;
                                }
                        }
                }
        out_values[x][y]=m[5];
        y=y+1;
        }
        while (y <=(column-1));
        x=x+1;
        }
        while (x<=(row-1));

        for (i=0;i<m_size;i++)
        for(j=0;j<m_size;j++)
        {
         values[i][j]=out_values[i][j];
        }

 }

int add_images (values,values2,m_size)
short int values[256][256],values2[256][256],m_size;
{
        int max_value,i,j;
        max_value=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                values[i][j]=values[i][j]+values2[i][j];
                if (values[i][j] > 255)
                {values[i][j]=255;}
         }
}

int subtract_images (values,values2,m_size)
short int values[256][256],values2[256][256],m_size;
{
        int max_value,i,j;
        max_value=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                values[i][j]=values[i][j]-values2[i][j];
                if (values[i][j] < 0)
                {values[i][j]=0;}
         }
}
```

```
/********** Function to determine corners of rectangular roi **********/
int region_of_interest (temp_in,temp_out,ct)
short ct[100],temp_in[256][256],temp_out[256][256];
{
        short *buf2;
        short *fpoint;
        int handle, bytes,i,j;
        short scale;
        short display_size;
        char roi_file[50];
        buf2=(short *) calloc(300,1);

        printf("Enter name of roi file:> ");
        scanf("%24s",roi_file);
        if ((handle=open(roi_file,O_RDONLY ))==-1) {
                printf("Error opening file.\n");
                exit(1);
        }
        if((bytes=read(handle,buf2,200))==-1) {
                printf("Read failed");
                exit(1);
        }
         else {
                printf("Read: %d bytes read.\n",bytes);
        }

 /* ** Get information from MUMC ROI file ** */

        fpoint=buf2;
        for (i=0; i< 9; i++)
                {fpoint++;}
        display_size=*fpoint;
        fpoint++;
        for (i=0; i< 30; i++){
                ct[i]= *fpoint;
                fpoint++;
        }
        close(handle);
        scale=display_size/256;
        ct[0]=ct[0]/scale;
        ct[1]=(512-ct[1])/scale;
        ct[2]=ct[2]/scale;
        ct[3]=(512-ct[3])/scale;

/* Set outside roi to 0 */
        for(i=ct[1]; i<=ct[3]; i++)
                for(j=ct[0]; j<=ct[2]; j++){
                        temp_out[i][j]=temp_in[i][j];
                }
}

/************ Function to find histogram of image matrix **********/

int histogram(values,m_size,bin_size)
short int values[256][256],m_size,bin_size;
{
  int i,j,bin,max_ct;
  short int histo[256],h[256],slope[256],max[20],ct=0;

        for (i=0;i<m_size;i++)
        for(j=0;j<m_size;j++)
        {
                if(values[i][j] !=0)
                {bin=values[i][j]/bin_size;
                histo[bin]=histo[bin]+1; }
```

```
        }
 /* Apply a 5-point smooth to histogram */
        for(j=0;j<m_size;j++)
        {
     h[j]=(histo[j]+2*histo[j+1]+3*histo[j+2]+2*histo[j+3]+histo[j+4])/9;
        }
        for(i=4; i<m_size; i++)
                {slope[i]=h[i+1]-h[i];}
        for(i=4; i<256; i++)
        {
        if(slope[i]>0 && slope[i+1]<0)
                {max[ct]=i+2;
                ct++;}
        }
        max_ct=max[0];
        return(max_ct);
/*printf("max= %d\n",max[0]);*/
}

/************ Function to find the maximum value of an array ************/

int maximum (values,m_size,values2)
short int values[256][256],m_size,values2[5];
{
        int max_value,i,j;
        max_value=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++){
                if (values[i][j] > max_value) {
                        max_value=values[i][j];
                        values2[0]=i; values2[1]=j;
                        values2[2]=max_value;
                }
        }
        /*return(max_value,xm,ym);*/
}

/****** Function to optimize contrast using histogram equalization *****/
int optimize_contrast(values,m_size,rx1,rx2,ry1,ry2,max_pixel)
short int values[256][256],m_size,rx1,rx2,ry1,ry2,max_pixel;
{
short int hist[12500];
int npix,i,j;
for (i=0;i< 12500;i++)
        hist[i]=0;

for (i=ry1;i<=ry2;i++)
        for(j=rx1;j<=rx2;j++)
        {
                hist[values[i][j]]++;
        }
        /*npix = m_size*(m_size-1) - hist[0];*/
        npix= ((rx2-rx1)*(ry2-ry1))-hist[0];
        hist[0] = 0;

/* max_pixel=maximum(values,m_size);*/
printf("\n Maximum ct#=%d\n", max_pixel);
printf("\n npix= %d\n",npix);
        for (i = 1; i < max_pixel; i++)
                hist[i] += hist[i-1];

        for (i=0;i<m_size;i++)
        for(j=0;j<m_size;j++)
                {
                values[i][j] = (hist[values[i][j]] * max_pixel/npix);
                }
```

```
}

/************ Function to find the mean value of an array ************/

int mean (values,m_size)
short int values[256][256],m_size;
{
        int mean_value,sum,mean_counter,i,j;
        sum=0;
        mean_counter=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                if (values[i][j] !=0)
                { sum=values[i][j]+sum;
                 mean_counter++;}
         }
        mean_value=sum/mean_counter;
        return(mean_value);
}

/******************** Function to threshold an image ***************/
/* Values above threshold remain the same, below T set to 0 */

int thresholding  (values,values2,m_size,T)
short int values[256][256],values2[256][256],m_size,T;
{
        int i,j,counter;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                if ((values2[i][j] > T) || (values[i][j] > 100 ))
                        {values[i][j]=255;
                        counter++;}
                else
                        {values[i][j]=0;}
        }
}


/************ Function to determine the moments of image ************/
int moment_of_inertia(values,temp_image,m_size,rx1,rx2,ry1,ry2,mom,max_pixel,res)
FILE *res;
short int values[256][256],temp_image[256][256],m_size,rx1,rx2,ry1,ry2,max_pixel;
 int mom[5][5];
{
        int mean_value,total_sum,x,y;
        int i,j,f,b_term,x_mean,y_mean,n_pixels;
        float sum00,sum01,sum10,sum11,sum20,sum02,sum_x,sum_y,x_m,y_m,sum_weight;
        float sum_mask,tan_theta1,tan_theta2,f_dec,weight;
        float IX_CRT_A, IY_CRT_A, IXY_CRT_A;
        float IX_TOT_W, IY_TOT_W, IXY_TOT_W;
        float IX_TOT, IY_TOT, IXY_TOT;
        sum00=0;
        sum10=0; sum01=0;
        for (x=0; x< 6; x++)
                for (y=0; y< 6; y++)
                        {mom[x][y]=0;}

        sum_x=0.0; sum_y=0.0; n_pixels=0.0;
        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                        if ((values[x][y] !=0)&&(values[x][y] >=900)){
                                n_pixels++;
                                sum_x= (float)x+sum_x;
                                sum_y=(float)y+sum_y;
```

```
                        }
                }
        x_m=sum_x/(float)n_pixels; y_m=sum_y/(float)n_pixels;
        printf("centre of mass for cortical shell(non-weighted) %.2f  %.2f\n",x_m,y_m);

/* Calculate moments around X, Y, and XY axis for CORTICAL BONE*/
        IX_CRT_A=0.0;
        IY_CRT_A=0.0;
        IXY_CRT_A=0.0;
        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                        if ((values[x][y] !=0)&&(values[x][y] >=900)){  /* DEFINE PURE CORT
                                IX_CRT_A=IX_CRT_A+(0.1089)*(y_m-(float)y)*(y_m-(float)y)*(0
                                IY_CRT_A=IY_CRT_A+(0.1089)*(x_m-(float)x)*(x_m-(float)x)*(0
                                IXY_CRT_A=IXY_CRT_A+(0.1089)*(x_m-(float)x)*(y_m-(float)y)*
                        }
                }
fprintf(res,"   Moment of Inertia .... CORTICAL SHELL\n");
fprintf(res,"           About X-axis: %.3f mm4\n",IX_CRT_A);
fprintf(res,"           About Y-axis: %.3f mm4\n",IY_CRT_A);
fprintf(res,"           About XY-axis: %.3f mm4\n",IXY_CRT_A);
fprintf(res,"\n");
fprintf(res,"\n");

/* Calculate moments around X, Y, and XY axis for TOTAL BONE CROSS-SECTION  (UNWEIGHTED)*/

        /*printf("Attenuation values will be weighted by %d\n",max_pixel);*/
        sum_x=0.0; sum_y=0.0; n_pixels=0;
        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                        if ((values[x][y] !=0)&&(values[x][y] >=400)){
                                weight=1.0;
                                n_pixels++;
                                sum_x=weight*(float)x+sum_x;
                                sum_y=weight*(float)y+sum_y;
                        }
                }
        x_m=sum_x/(float)n_pixels; y_m=sum_y/(float)n_pixels;
        printf("centre of mass of total bone cross-section(unweighted) %.2f  %.2f\n",x_m,y_
        IX_TOT=0.0; IY_TOT=0.0; IXY_TOT=0.0;
        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                        if ((values[x][y] !=0)&&(values[x][y] >=400)){  /* DEFINE TOTAL BON
                                weight=1.0;
                                IX_TOT=IX_TOT+weight*(0.1089)*(y_m-(float)y)*(y_m-(float)y)
                                IY_TOT=IY_TOT+weight*(0.1089)*(x_m-(float)x)*(x_m-(float)x)
                                IXY_TOT=IXY_TOT+weight*(0.1089)*(x_m-(float)x)*(y_m-(float)
                        }
                }
fprintf(res,"   Moment of Inertia .... WHOLE BONE AREA(non-weighted)\n");
fprintf(res,"           About X-axis: %.3f mm4\n",IX_TOT);
fprintf(res,"           About Y-axis: %.3f mm4\n",IY_TOT);
fprintf(res,"           About XY-axis: %.3f mm4\n",IXY_TOT);
fprintf(res,"\n");
fprintf(res,"\n");

/* Calculate moments around X, Y, and XY axis for TOTAL BONE CROSS-SECTION  (WEIGHTED)*/
        max_pixel=1200;
        sum_x=0.0; sum_y=0.0; n_pixels=0; sum_weight=0.0;
        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                        if ((values[x][y] !=0)&&(values[x][y] >=400)){
                                weight=(float) values[x][y]/(float) max_pixel;
                                n_pixels++;
                                sum_weight=weight+sum_weight;
                                sum_x=weight*(float)x+sum_x;
```

```
                                sum_y=weight*(float)y+sum_y;
                        }
                }
        x_m=sum_x/sum_weight; y_m=sum_y/sum_weight;
        printf("centre of mass of total bone cross-section(weighted) %.2f  %.2f\n",x_m,y_m)
        IX_TOT_W=0.0; IY_TOT_W=0.0; IXY_TOT_W=0.0;
        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                        if ((values[x][y] !=0)&&(values[x][y] >=400)){  /* SELECT TOTAL BON
                                weight=(float) values[x][y]/(float)max_pixel;
                                IX_TOT_W=IX_TOT_W+weight*(0.1089)*(y_m-(float)y)*(y_m-(floa
                                IY_TOT_W=IY_TOT_W+weight*(0.1089)*(x_m-(float)x)*(x_m-(floa
                                IXY_TOT_W=IXY_TOT_W+weight*(0.1089)*(x_m-(float)x)*(y_m-(fl
                        }
                }

 fprintf(res,"   Moment of Inertia .... WHOLE BONE AREA(weighted)\n");
 fprintf(res,"          About X-axis: %.3f mm4\n",IX_TOT_W);
 fprintf(res,"          About Y-axis: %.3f mm4\n",IY_TOT_W);
 fprintf(res,"          About XY-axis: %.3f mm4\n",IXY_TOT_W);
 fprintf(res,"\n");
 fprintf(res,"\n");

 }


 /********** Function to remove isolated points  ***********/
 int clean_image (values,m_size)
 short int values[256][256],m_size;
 {
        int i,j,m[9];
        for (i=2; i< m_size-1; i++)
        for (j=2;j< m_size-1; j++)
         {
         m[0]=values[i-1][j-1];
         m[1]=values[i-1][j];
         m[2]=values[i-1][j+1];
         m[3]=values[i][j-1];
         m[4]=values[i][j];
         m[5]=values[i][j+1];
         m[6]=values[i+1][j-1];
         m[7]=values[i+1][j];
         m[8]=values[i+1][j+1];
if((m[4]==1)&&(m[0]==1)&&(m[1]==1)&&(m[2]==1)&&(m[3]==0)&&(m[5]==0)&&(m[6]==0)&&(m[7]==0)&&
                        {values[i][j]=0;}
if((m[4]==1)&&(m[0]==0)&&(m[1]==0)&&(m[2]==0)&&(m[3]==0)&&(m[5]==0)&&(m[6]==0)&&(m[7]==0)&&
                        {values[i][j]=0;}
if((m[4]==0)&&(m[0]==1)&&(m[1]==1)&&(m[2]==1)&&(m[3]==1)&&(m[5]==1)&&(m[6]==1)&&(m[7]==1)&&
                        {values[i][j]=1;}

        }
}

/****************** Function to grow region given a seed pixel *************/
int stack=0;
short int *pstack;

int grow_region (pimage,im_rows,im_columns,xc,yc,t)
short *pimage;
int im_rows,im_columns;
int xc,yc;
short t;
{
   int i,j,x,y;
    short *pbuf;
```

```
    pstack = (short int *) malloc(32000);
    if (pstack==NULL)
         { printf("\nGrow region: Not enough memory\n");  return(-1); }
;
    px_push(xc,yc);
    *(pimage+(xc*im_columns)+yc)=3000;

    while(stack)
         {
           px_pop(&x,&y);

          i=(x-1)*im_columns; j=y;
           if (*(pimage+i+j)==3000) i=i;
             else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                          px_push(x-1,y);      }
                  else *(pimage+i+j)=0;

          i=(x+1)*im_columns; j=y;
          if (*(pimage+i+j)==3000) i=i;
            else if (*(pimage+i+j)>=t) {*(pimage+i+j)=3000;
                                          px_push(x+1,y);      }
                 else *(pimage+i+j)=0;

          i=x*im_columns; j=y-1;
          if (*(pimage+i+j)==3000) i=i;
            else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                          px_push(x,y-1);      }
                 else *(pimage+i+j)=0;

          i=x*im_columns; j=y+1;
          if (*(pimage+i+j)==3000) i=i;
            else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                          px_push(x,y+1);      }
                 else *(pimage+i+j)=0;

/* continue for 8-connectivity */

          i=(x+1)*im_columns; j=y-1;
          if (*(pimage+i+j)==3000) i=i;
            else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                          px_push(x+1,y-1);    }
                 else *(pimage+i+j)=0;

          i=(x+1)*im_columns; j=y+1;
          if (*(pimage+i+j)==3000) i=i;
            else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                          px_push(x+1,y+1);    }
                 else *(pimage+i+j)=0;


          i=(x-1)*im_columns; j=y-1;
          if (*(pimage+i+j)==3000) i=i;
            else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                          px_push(x-1,y-1);    }
                 else *(pimage+i+j)=0;

          i=(x-1)*im_columns; j=y+1;
          if (*(pimage+i+j)==3000) i=i;
            else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                          px_push(x-1,y+1);    }
                 else *(pimage+i+j)=0;

        }


   x=0;
```

```
    for (i=0;i<im_rows*im_columns;i++)
          {      if (*(pimage+i)!=3000) *(pimage+i)=0;
                 else  x++;     }
   free(pstack);
   return(x);
  }

  int px_push(x,y)
  int x;
  int y;
  {
    *pstack= (short int)x;
    pstack++;
    *pstack= (short int)y;
    pstack++;
    stack++;
  /* printf("\n %d  (%d,%d)",stack, x,y);*/
  }

  int px_pop(px, py)
  int *px;
  int *py;
  {
    short int i;
    pstack--;
    i=*pstack;
    *py=(int)i; pstack--;
    i=*pstack;
    *px=(int)i;
    stack--;
  }


  /*************** Function to extract cortical bone  *****************/

  int get_cortex(values,contour,m_size,rx1,rx2,ry1,ry2,T)
  short int values[256][256],contour[256][256],m_size,rx1,rx2,ry1,ry2,T;
  {
          int i,j,p1,p2,p3,p4,start_pixel=0;
          int delta,run_length,threshold;
          int temp[256][256];

  /* clear temp */
          for (i=0; i< m_size; i++){
                  for (j=0;j< m_size; j++){
                          temp[i][j] =0;
                  }

          }
  printf("Extracting cortex .....\n");

           for (i=ry1; i<=ry2; i++){
                  for(j=rx1; j<=rx2; j++){
                          p1=values[i][j];
                          p2=values[i][j+1];
                          p3=values[i-1][j+1];
                          p4=values[i][j-1];
                          if ((p1>T)&&(p2 > T || p3>T || p4>T)){
                                          temp[i][j]=255;
                          }
                  }
          }

  /* save cortical shell **/

  for (i=0; i< m_size; i++)
```

```
        for (j=0;j< m_size; j++){
                values[i][j]=temp[i][j];
         }
}


/********************* Function to define bone contour ****************/
int contour(values,contour,m_size,rx1,rx2,ry1,ry2,T)
short int values[256][256],contour[256][256],m_size,rx1,rx2,ry1,ry2,T;
{
        int i,j,p1,p2,p3,p4,start_pixel=0;
        int delta,run_length,threshold;
        int temp[256][256];

/* clear contour */
        for (i=0; i< m_size; i++){
                for (j=0;j< m_size; j++){
                        contour[i][j] =0;
                        temp[i][j]=values[i][j];
                }

        }

printf("Extracting  outer contour .....\n");

/* define contour one pixel thick
 Scan along a row */
threshold=99;
i=ry1+2;
do {
        for(j=rx1+2; j<rx2-2; j++) {
        run_length=0.5*(rx1+rx2);
        if (j>run_length)
        {goto next_i;}

        delta=temp[i][j]-temp[i][j-1];
         if ((abs(delta)> threshold)) {
                /*contour[i][j-1]=255;*/
                contour[i][j]=255;
                goto next_i;
                }
        }
next_i:
i++;
}
while(i<ry2-2);

/* Scan down a column */
j=rx1+2;
do {
        for(i=ry1+2;i<ry2-2;i++) {
        run_length=0.5*(ry1+ry2);
        if (i>run_length)
        {goto next_j;}

        delta=temp[i][j]-temp[i-1][j];
         if ((abs(delta)> threshold)) {
                /* contour[i-1][j]=255;*/
                contour[i][j]=255;
                goto next_j;}
        }
next_j:
j++;
}
while(j<rx2-2);
```

```
   /* Scan back along a row */
   i=ry1+2;
   do {
           for(j=rx2-2; j>rx1+2; j--)
           {
           run_length=0.5*(rx1+rx2);
           if (j<run_length)
                   {goto next_x;}
           delta=temp[i][j]-temp[i][j-1];
            if ((abs(delta)> threshold)){
                   /* contour[i][j]=255;*/
                   contour[i][j-1]=255;
                   goto next_x;}
           }
  next_x:
  i++;
  }
  while(i<ry2-2);


   /*Scan back along a column */
  j=rx1+2;
  do {

          for(i=ry2-2; i>ry1+2; i--){
          run_length=0.5*(ry1+ry2);
          if (i<run_length)
                  {goto next_y;}
          delta=temp[i][j]-temp[i-1][j];
           if ((abs(delta)> threshold)){
                  /*contour[i][j]=255;*/
                  contour[i-1][j]=255;
                  goto next_y;}
          }
 next_y:
 j++;
 }
 while(j<rx2-2);
         printf("end of contour\n");

 }



 /*************** Function to define bone contour *****************/
 int contour2(values,contour,m_size,rx1,rx2,ry1,ry2,T)
 short int values[256][256],contour[256][256],m_size,rx1,rx2,ry1,ry2,T;
 {
         int i,j,p1,p2,p3,p4,start_pixel=0;
         int delta,run_length,threshold,l_contour=0;
         int temp[256][256];


 /* clear contour */
         for (i=0; i< m_size; i++){
                 for (j=0;j< m_size; j++){
                         contour[i][j] =0;
                         values[i][j]=temp[i][j];
                 }

         }

 printf("Extracting  outer contour .....\n");

 /* define contour one pixel thick
```

```
  Scan along a row */
 threshold=99;
 i=ry1+2;
 do {
         for(j=rx1+2; j<rx2-2; j++) {
         run_length=0.5*(rx1+rx2);
         if (j>run_length)
         {goto next_i;}

         delta=temp[i][j]-temp[i][j-1];
          if ((abs(delta)> threshold)) {
                 /*contour[i][j-1]=255;*/
                 contour[i][j]=255;
                 l_contour++;
                 goto next_i;
                 }
         }
 next_i:
 i++;
 }
 while(i<ry2-2);

 /* Scan down a column */
 j=rx1+2;
 do {
         for(i=ry1+2;i<ry2-2;i++) {
         run_length=0.5*(ry1+ry2);
         if (i>run_length)
         {goto next_j;}

         delta=temp[i][j]-temp[i-1][j];
          if ((abs(delta)> threshold)) {
                 /* contour[i-1][j]=255;*/
                 contour[i][j]=255;
                 l_contour++;
                 goto next_j;}
         }
 next_j:
 j++;
 }
 while(j<rx2-2);

 /* Scan back along a row */
 i=ry1+2;
 do {
         for(j=rx2-2; j>rx1+2; j--)
         {
         run_length=0.5*(rx1+rx2);
         if (j<run_length)
                 {goto next_x;}
         delta=temp[i][j]-temp[i][j-1];
          if ((abs(delta)> threshold)){
                 /* contour[i][j]=255;*/
                 contour[i][j-1]=255;
                 l_contour++;
                 goto next_x;}
         }
 next_x:
 i++;
 }
 while(i<ry2-2);


  /*Scan back along a column */
 j=rx1+2;
 do {
```

```
        for(i=ry2-2; i>ry1+2; i--){
        run_length=0.5*(ry1+ry2);
        if (i<run_length)
                {goto next_y;}
        delta=temp[i][j]-temp[i-1][j];
         if ((abs(delta)> threshold)){
                /*contour[i][j]=255;*/
                contour[i-1][j]=255;
                l_contour++;
                goto next_y;}
        }
next_y:
j++;
}
while(j<rx2-2);
        printf("end of contour");
return(l_contour);
}


/************** Function to  fill in contour given a seed pixel **************/
int stack2=0;
short int *pstack2;

int fill_contour(pimage2,im_rows,im_columns,xc,yc,t)
short *pimage2;
int im_rows,im_columns;
int xc,yc;
short t;
{
   int i,j,x,y;
    short *pbuf;
   pstack2 = (short int *) malloc(32000);
   if (pstack2==NULL)
        { printf("\nGrow region: Not enough memory\n");  return(-1); }

   px_push2(xc,yc);
   *(pimage2+(xc*im_columns)+yc)=3000;

   while(stack2)

        { px_pop2(&x,&y); i=(x-1)*im_columns; j=y;

          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) { *(pimage2+i+j)=3000;
                                          px_push2(x-1,y);     }
                else *(pimage2+i+j)=t;

          i=(x+1)*im_columns; j=y;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) {*(pimage2+i+j)=3000;
                                          px_push2(x+1,y);     }
                else *(pimage2+i+j)=t;

          i=x*im_columns; j=y-1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) { *(pimage2+i+j)=3000;
                                          px_push2(x,y-1);     }
                else *(pimage2+i+j)=t;

          i=x*im_columns; j=y+1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) { *(pimage2+i+j)=3000;
                                          px_push2(x,y+1);     }
                else *(pimage2+i+j)=t;
```

```
  /* continue for 8-connectivity */

          i=(x+1)*im_columns; j=y-1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j)<t) { *(pimage2+i+j)=3000;
                                         px_push2(x+1,y-1);  }
                else *(pimage2+i+j)=t;

          i=(x+1)*im_columns; j=y+1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j)<t) { *(pimage2+i+j)=3000;
                                         px_push2(x+1,y+1);  }
                else *(pimage2+i+j)=t;


          i=(x-1)*im_columns; j=y-1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j)<t) { *(pimage2+i+j)=3000;
                                         px_push2(x-1,y-1);  }
                else *(pimage2+i+j)=t;

          i=(x-1)*im_columns; j=y+1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j)<t) { *(pimage2+i+j)=3000;
                                         px_push2(x-1,y+1);  }
                else *(pimage2+i+j)=t;
        }

    x=0;
    for (i=0;i<im_rows*im_columns;i++)
          {     if (*(pimage2+i)!=3000 && *(pimage2+i)!=t) *(pimage2+i)=0;
                else  x++;    }
   free(pstack2);
   return(x);

}
int px_push2(x,y)
int x;
int y;
{
  *pstack2= (short int)x;
  pstack2++;
  *pstack2= (short int)y;
  pstack2++;
  stack2++;
/* printf("\n %d  (%d,%d)",stack2, x,y);*/
}

int px_pop2(px, py)
int *px;
int *py;
{
  short int i;
  pstack2--;
  i=*pstack2;
  *py=(int)i; pstack2--;
  i=*pstack2;
  *px=(int)i;
  stack2--;
}



/***************** Function to apply an adaptive threshold *****************/
```

```
int adaptive_t(in_values,values,m_size)
short values[256][256],in_values[256][256],m_size;
{
int i,j,k,x,y,mask=4,sf=9;
int sum1,sum2,sum3;
short values_x[256][256];

for (k=1; k<=mask; k++)
{
        x=2;
        do
        {
        y=2;
        do
        {
        sum1=(values[x-1][y-1])+(values[x-1][y])+(values[x-1][y+1]);
        sum3=(values[x+1][y-1])+(values[x+1][y])+(values[x+1][y+1]);
        values_x[x][y]=(sum1+sum2+sum3)/sf;
       if (values_x[x][y] > 32000 )
                values_x[x][y]=0;
        if (values_x[x][y] < 0)
        {
                values_x[x][y]=0;
        }
        y=y+1;
        }
        while (y <=(m_size-1));
        x=x+1;
        }
        while (x<=(m_size-1));

        for (i=0; i<m_size; i++)
        for (j=0; j<m_size; j++)
                {
                values[i][j]=values_x[i][j];
                }
}
/* apply an adaptive threshold to create a binary image */
        for (i=0;i<m_size;i++)
        for(j=0;j<m_size;j++)
        {
         if ( in_values[i][j] > values_x[i][j])
                values[i][j]=1;
         else
                values[i][j]=0;
        }
}

/**** Function  to implement the Euclidean distance transform ****/

int distance_transform(values,m_size)
short int values[256][256],m_size;
{
        int e1,e2,e3,e4,e5,min_pixel=0;
        int i,j;

/*** Apply Euclidean (2-3)distance transform  ***/
/** Forward raster scan **/

        i=2;
        do {
           j=2;
           do  {
                e1=2+values[i][j-1];
                min_pixel=e1;
                e2=3+values[i-1][j-1];
```

```
                if (e2 < min_pixel) {
                        min_pixel=e2;
                }
                e3=2+values[i-1][j];
                if (e3 < min_pixel) {
                        min_pixel=e3;
                }
                e4=3+values[i-1][j+1];
                if (e4 < min_pixel) {
                        min_pixel=e4;
                }
                e5=values[i][j];
                if (e5 == 0) {
                        min_pixel=0;
                }
                values[i][j]=min_pixel;
                j=j+1;
          }
          while (j <= m_size-1);
        i=i+1;
        }
        while (i <= m_size-1);


/*** Reverse raster scan ***/

        i=m_size-1;
        do {
        j=m_size-1;
          do  {
                e1=2+values[i][j+1];
                min_pixel=e1;
                e2=3+values[i+1][j+1];
                if (e2 < min_pixel) {
                        min_pixel=e2;
                }
                e3=2+values[i+1][j];
                if (e3 < min_pixel) {
                        min_pixel=e3;
                }
                e4=3+values[i+1][j-1];
                if (e4 < min_pixel) {
                        min_pixel=e4;
                }
                e5=values[i][j];
                if (e5  < min_pixel) {
                        min_pixel=e5;
                }
                values[i][j]=min_pixel;
                j=j-1;
          }
          while (j >= 2);
        i=i-1;
        }
        while (i >= 2);

}

/* Routine to determine the number of regions connected by (i,j) */

int crossing_index(values,ii,jj)
short int values[256][256];
int ii,jj;
{
        int i,j,count;
        short int k;
```

```
        count=0;
  /* start at position 8 */
        i=ii-1; j=jj-1; k=values[i][j];

  /* move clockwise around (ii,jj), counting level changes */

        j++; /* move to (i-1,j) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        j++; /* move to (i-1,j+1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        i++; /* move to (i,j+1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        i++; /* move to (i+1,j+1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        j--; /* move to (i+1,j) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        j--; /* move to (i+1,j-1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        i--; /* move to (i,j-1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        i--; /* move to (i-1,j-1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        return count/2;
}


/* Routine to determine the number of 8-nearest neigbbours to (i,j) */

int nay8(values,i,j,val)
short int values[256][256];
int i,j,val;
{
        int k;

        k=0;
        if(i < 1 || i >=255) return 0;
        if(j < 1 || j >=255) return 0;

        if(values[i][j]!= val) return 0;
        if(values[i-1][j]==val) k++;
        if(values[i-1][j+1]==val) k++;
        if(values[i][j+1]==val) k++;
        if(values[i+1][j+1]==val) k++;
        if(values[i+1][j]==val) k++;
        if(values[i+1][j-1]==val) k++;
        if(values[i][j-1]==val) k++;
        if(values[i-1][j-1]==val) k++;

        return k;
}


/* Zhang-Suen type of thinning procedure. Thin region labelled VAL */

int thinzs(values, val)
short int values[256][256];
int val;
{
        int i,j,n,again,bg;
        short int values2[256][256];

bg=0;

/* clear second image space to 0 */
        for(i=0; i<256; i++)
```

```
        {
                for(j=0; j<256; j++)
                        {values2[i][j]=0;}
        }

/* copy original image to second image space */

        for(i=0; i<256; i++)
        {
                for(j=0; j<256; j++)
                        {values2[i][j]=values[i][j];}
        }

do{

/* first pass through image */
        again=0;
        for(i=0; i<256; i++){
                for(j=0; j<256; j++)  {
                        if(values2[i][j]!=val) continue;
                        n=nay8(values2,i,j,val);
                        if( (n >=2 ) && (n <=6))  {
                          if(crossing_index(values2,i,j)==1)  {
                            if((values2[i-1][j]==bg)||
                                        (values2[i][j+1]==bg)||
                                        (values2[i+1][j]==bg)) {
                                if((values2[i][j+1]==bg)||
                                        (values2[i+1][j]==bg)||
                                        (values2[i][j-1]==bg)) {
                                values[i][j]=bg;
                                again=1;
                                }
                             }
                           }
                        }
                        }
        }

        /* copy original image to second image space */

        for(i=0; i<256; i++)
                for(j=0; j<256; j++)
                        {values2[i][j]=values[i][j];}


/* second pass through image */

        for(i=0; i<256; i++){
                for(j=0; j<256; j++)  {
                        if(values[i][j]!=val) continue;
                        n=nay8(values,i,j,val);
                        if( (n >=2 ) && (n <=6))  {
                          if(crossing_index(values,i,j)==1)  {
                            if((values[i-1][j]==bg)||
                                        (values[i][j+1]==bg)||
                                        (values[i][j-1]==bg)) {
                                if((values[i-1][j]==bg)||
                                        (values[i+1][j]==bg)||
                                        (values[i][j-1]==bg)) {
                                values2[i][j]=bg;
                                again=1;
                                }
                             }
                           }
                        }
                }
```

```
                        }
        }


        /* copy second image to original image space */

        for(i=0; i<256; i++)
        {
                for(j=0; j<256; j++)
                        {values[i][j]=values2[i][j];}
        }
} while (again);

}



/************** Function to check for node points  *************/
int indices(in_values,values,rows,columns,val,nd_label,fe_label,res_file)
short values[256][256],in_values[256][256];
int rows,columns,nd_label,fe_label,val;
FILE *res_file;
{
int x,y,node_sum,node=0,free_ends=0;
int i,j,isolated_points=0,network_length=0,m[9];
int k,count;
float CI;

for(i=0; i<=9; i++)
        m[i]=0;      /* clear m to 0 */

for (i=0; i<rows;i++)
        for (j=0; j<columns;j++){
                values[i][j]=0;    /* clear values to 0 */
                if (in_values[i][j]==val){
                        values[i][j]=100;
                        network_length++;
                }
        }

for (x=2; x<=(rows-1); x++)
        for (y=2; y <= (columns-1); y++) {
/* first make sure that two adjacent points are not both identifies as nodes */
                if (values[x][y]==0) continue;
                if (values [x-1][y-1]==nd_label) continue;
                if (values[x-1][y]==nd_label) continue;
                if(values[x-1][y+1]==nd_label) continue;
                if (values[x][y-1]==nd_label) continue;
                if(values[x][y+1]==nd_label) continue;
                if (values[x+1][y-1]==nd_label) continue;
                if (values[x+1][y]==nd_label) continue;
                if (values[x+1][y+1]==nd_label) continue;

/* start at position 8 */
                count=0; i=x-1; j=y-1; k=values[i][j];
/* move clockwise around (ii,jj), counting level changes */

                j++; /* move to (i-1,j) */
                if (k !=values[i][j] ){k=values[i][j]; count++;}
                j++; /* move to (i-1,j+1) */
                if (k !=values[i][j] ){k=values[i][j]; count++;}
                i++; /* move to (i,j+1) */
                if (k !=values[i][j] ){k=values[i][j]; count++;}
                i++; /* move to (i+1,j+1) */
                if (k !=values[i][j] ){k=values[i][j]; count++;}
                j--; /* move to (i+1,j) */
```

```
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    j--; /* move to (i+1,j-1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    i--; /* move to (i,j-1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    i--; /* move to (i-1,j-1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}

                    if(count >=6) {
                            values[x][y]=nd_label; /* label nodes  */
                            node++;
                    }
            }

    /* Score for free ends and isolated points */
    node_sum=0;
    for (x=2; x<=(rows-1); x++)
            for (y=2; y <= (columns-1); y++) {
                    m[0]=in_values [x-1][y-1];
                    m[1]=in_values[x-1][y];
                    m[2]=in_values[x-1][y+1];
                    m[3]=in_values[x][y-1];
                    m[4]=in_values[x][y];
                    m[5]=in_values[x][y+1];
                    m[6]=in_values[x+1][y-1];
                    m[7]=in_values[x+1][y];
                    m[8]=in_values[x+1][y+1];
                    node_sum=m[0]+m[1]+m[2]+m[3]+m[5]+m[6]+m[7]+m[8];
                            if((m[4]>0)&&(node_sum==val)){
                                    values[x][y]=fe_label; /* label free ends  */
                                    free_ends++;
                            }
                            if((m[4]>0)&&(node_sum==0)) {
                                    isolated_points++;
                            }
                            if((m[4]>0)&&(m[5]==val)&&(m[7]==val)&&(m[8]==val)) {
                                    values[x][y]=nd_label; /* label nodes  */
                                    node++;
                            }

            }
CI=(((float)node-(float)free_ends-(float)isolated_points)/(float)network_length)*100.0;

printf("Network length= %d\n",network_length);
printf("Nodes= %d\n",node);
printf("Free ends= %d\n",free_ends);
printf("Isolated points=%d\n",isolated_points);
printf("Connectivity Index= %.2f\n", CI);
fprintf(res_file,"  \n");
fprintf(res_file,"Network length= %d\n",network_length);
fprintf(res_file,"Nodes= %d\n",node);
fprintf(res_file,"Free ends= %d\n",free_ends);
fprintf(res_file,"Isolated points=%d\n",isolated_points);
fprintf(res_file,"Connectivity Index= %.2f\n", CI);
  return(node);
}

/************* Function to check that contour is closed  ************/
int contour_check(in_values,values,rows,columns,val,fe_label)
short values[256][256],in_values[256][256];
int rows,columns,fe_label,val;
{
int x,y,node_sum,node=0,free_ends=0;
int i,j,isolated_points=0,network_length=0,m[9];
int k,count;
```

```
  for(i=0; i<=9; i++)
          m[i]=0;      /* clear m to 0 */

  /* Score for free ends */
  node_sum=0;
  for (x=2; x<=(rows-1); x++)
          for (y=2; y <= (columns-1); y++) {
                  values[x][y]=0;
          }

  for (x=2; x<=(rows-1); x++)
          for (y=2; y <= (columns-1); y++) {
                   m[0]=in_values [x-1][y-1];
                   m[1]=in_values[x-1][y];
                   m[2]=in_values[x-1][y+1];
                   m[3]=in_values[x][y-1];
                   m[4]=in_values[x][y];
                   m[5]=in_values[x][y+1];
                   m[6]=in_values[x+1][y-1];
                   m[7]=in_values[x+1][y];
                   m[8]=in_values[x+1][y+1];
                   node_sum=m[0]+m[1]+m[2]+m[3]+m[5]+m[6]+m[7]+m[8];
                          if((m[4]>0)&&(node_sum==val)){
                                  values[x][y]=fe_label; /* label free ends   */
                                  free_ends++;
                          }
          }
    return(free_ends);
  }


  /*** Routine to count holes in binary image  ***/

  int hole_counter(h,values,rx1,rx2,ry1,ry2)
  short int values[256][256],rx1,rx2,ry1,ry2;
  int h[500];
  {
  int i,j,count=0,val=1;


  for(i=0; i<500; i++)
          h[i]=0;      /* clear h to 0 */

  for (i=ry1+2; i<ry2;i++)
          for (j=rx1+2 ; j<rx2;j++){
                  if (values[i][j] !=0)  continue;
                          /* printf("x and y= %d  %d\n", i,j);     */
                          h[count]=fill_hole (values,256,256,i,j,0,val);
                          /* printf("count= %d\n",count);
                          printf("area of hole= %d\n",h[count]);
                          printf("seeds were %d %d\n",i,j); */

                          count++;
                          val +=1;
          }
  return count;
  }


  /******* Function fill hole by region grow given a seed pixel **********/

  int stack3=0;
  short int *pstack3;

  int fill_hole (pimage,im_rows,im_columns,xc,yc,t,val)
  short *pimage;
```

```
  int im_rows,im_columns;
  int xc,yc;
  short t,val;
 {
     int i,j,x,y;
      short *pbuf;

     pstack3 = (short int *) malloc(32000);
     if (pstack3==NULL)
          { printf("\nGrow region: Not enough memory\n");  return(-1); }

     px_push3(xc,yc);
     *(pimage+(xc*im_columns)+yc)=val;

     while(stack3)
          {
            px_pop3(&x,&y);

           i=(x-1)*im_columns; j=y;
            if (*(pimage+i+j)==val) i=i;
              else if (*(pimage+i+j)<=t) { *(pimage+i+j)=val;
                                           px_push3(x-1,y);     }
                   else *(pimage+i+j)=*(pimage+i+j);

            i=(x+1)*im_columns; j=y;
            if (*(pimage+i+j)==val) i=i;
              else if (*(pimage+i+j)<=t) {*(pimage+i+j)=val;
                                           px_push3(x+1,y);     }
                   else *(pimage+i+j)=*(pimage+i+j);

            i=x*im_columns; j=y-1;
            if (*(pimage+i+j)==val) i=i;
              else if (*(pimage+i+j)<=t) { *(pimage+i+j)=val;
                                           px_push3(x,y-1);     }
                   else *(pimage+i+j)=*(pimage+i+j);

            i=x*im_columns; j=y+1;
            if (*(pimage+i+j)==val) i=i;
              else if (*(pimage+i+j)<=t) { *(pimage+i+j)=val;
                                           px_push3(x,y+1);     }
                   else *(pimage+i+j)=*(pimage+i+j);
          }


    x=0;
    for (i=0;i<im_rows*im_columns;i++)
          {      if (*(pimage+i)!=val) *(pimage+i)=*(pimage+i);
                 else  x++;     }
  free(pstack3);
  return(x);
}

int px_push3(x,y)
int x;
int y;
{
  *pstack3= (short int)x;
  pstack3++;
  *pstack3= (short int)y;
  pstack3++;
  stack3++;
 /*printf("\n %d  (%d,%d)",stack3, x,y);*/
}

int px_pop3(px, py)
int *px;
```

```
  int *py;
  {
    short int i;
    pstack3--;
    i=*pstack3;
    *py=(int)i; pstack3--;
    i=*pstack3;
    *px=(int)i;
    stack3--;
  }


  /****** Function to calculate mean and standard deviation of an array *****/
  int mean_std(values,m,std,count,res_file)
  int values[500],count;
  float m, std;
  FILE *res_file;
  {
  int i,sum=0;
  double sum_std=0.0;
  for (i=1; i<= count; i++)  /* calculate mean */
          {sum +=values[i];}
  m=(sum/((float)count))*0.33*0.33;
  printf("\n MEAN =%.2f mm2\n", m);
  fprintf(res_file,"\nMean hole size=%.2f mm2\n", m);

  for (i=1;i<=count;i++) {          /* calculate standard deviation */
          sum_std=sum_std+((values[i]-m)*(values[i]-m));}
          sum_std=sum_std/(count-1);
          std=sqrt(sum_std);
          /*printf("STD =%f\n", std);*/
  }


  /****** Function to sort an array of integers in ascending order *******/
  int sort(a,n,res_file)
  int a[500],n;
  FILE *res_file;
  {
          int i,j,temp;
          float min,max,median;
          for (i=0; i<=(n-1); ++i)
                  for (j=i+1; j<=n; ++j) {
                          if (a[i]>a[j]) {
                                  temp=a[i];
                                   a[i]=a[j];
                                   a[j]=temp;
                          }
                  }
          min=(float) a[0]*0.33*0.33;
          max=(float) a[n-1]*0.33*0.33;
          median=(float) a[n/2]*0.33*0.33;
          printf("Minimum= %.2f mm2\n", min);
          printf("Maximum= %.2f mm2\n", max);
          printf("Median= %.2f mm2\n", median);
          fprintf(res_file,"Median= %.2f mm2\n", median);
          fprintf(res_file,"Minimum= %.2f mm2\n", min);
          fprintf(res_file,"Maximum= %.2f mm2\n", max);

  }


  /* morphological operations */
```

```
int dilation(pimage, n_bytes, im_rows,im_columns, pelem,elem_rows,elem_columns)
int n_bytes, im_rows, im_columns, elem_rows, elem_columns;
unsigned char *pimage, *pelem;
{  int i,j,x,y, h_rows, h_col, sum, itemp, total, el_temp, im_temp, value;
   short int  ival, *pbi, *pimage2, *pi;
   unsigned char cval, *pbuf, *pcimage2, *pci, ctemp;

   if (n_bytes==1) pbuf = pimage;
        else if (n_bytes==2) pbi = (short int *) pimage;
          else return(-2);

   total=im_rows*im_columns;  ival=0; cval=0; j=0;
   do { if (n_bytes==2) ival=*pbi++;
              else cval=*pbuf++;  j++; } while(!ival && !cval && j<total);
/* printf("\n first no-zero value at %d  (%d  %d)", j, im_rows,im_columns); */
   if (!ival && !cval) return(-3);

   total=im_rows*im_columns*n_bytes;
   pcimage2=(unsigned char *) malloc(total);     /* get memory for new image  */
   if (pcimage2==NULL) return(-1);
   pimage2=(short int *) pcimage2;

   pbuf=pcimage2; for(i=0;i<total;i++)  *pbuf++=0;

   h_rows=elem_rows/2; h_col=elem_columns/2;

   pbi = (short int *) pimage;
   x=h_rows;
   do { y=h_col;
        do { sum=0;
             pbuf=pelem;
             for (i=0;i<elem_rows;i++)
                for (j=0;j<elem_columns;j++)
                   { el_temp=(int) (*pbuf); pbuf++;
                     if (n_bytes==2)
                             im_temp=(int) (*(pbi+(x-h_rows+i)*im_columns
                                              +y-h_col+j) );
                        else  im_temp=(int) (*(pimage+(x-h_rows+i)*im_columns
                                              +y-h_col+j) );
                     itemp = el_temp*im_temp;
                     sum += itemp;  }

             if (sum) {  if (n_bytes==2)
                                *(pimage2+x*im_columns+y) = ival;
                           else
                                *(pcimage2+x*im_columns+y) = cval;
                       }
             y++;
           } while (y < im_columns-h_col);
        x++;
      }  while (x < im_rows-h_rows);

  pbuf=pimage;  pci=pcimage2; j=0;
  pbi=(short int *) pimage; pi=pimage2;
  total=im_rows*im_columns*n_bytes;
  for(i=0;i<total;i++) *pbuf++ = *pci++;

/*    { if (n_bytes==2)
        { if (*pi && !j)
           { printf(" \n first no zero at  %d (%d)  ", (int)*pci, i);  j=1; }
          *pbi++=*pi++;  }
      else
        { if (*pci && !j)
           { printf(" \n first no zero at  %d (%d)  ", (int)*pci, i);  j=1; }
          *pbuf++ = *pci++;  }
   }
```

```
 */
    value=ival + (int)cval;
    free(pcimage2);
    return(value);
}



int erosion(pimage, n_bytes, im_rows,im_columns, pelem,elem_rows,elem_columns)
int n_bytes, im_rows, im_columns, elem_rows, elem_columns;
unsigned char  *pimage, *pelem;
{  int i,j,x,y, h_rows, h_col, sum, itemp, total, el_temp, im_temp, value;
   int el_sum;
   short int  ival, *pbi, *pimage2;
   unsigned char cval, *pbuf, *pcimage2, *pci, ctemp;

   if (n_bytes==1) pbuf = pimage;
        else if (n_bytes==2) pbi = (short int *) pimage;
          else return(-2);

   i=im_rows*im_columns;  ival=0; cval=0;
   do { if (n_bytes==2) ival=*pbi++;
               else cval=*pbuf++; i--; } while(!ival && !cval && i);
   if (!ival && !cval) return(-3);
   value=ival + (int)cval;

   total=im_rows*im_columns*n_bytes;
   pcimage2=(unsigned char *) malloc(total);     /* get memory for new image  */
   if (pcimage2==NULL) return(-1);
   pimage2=(short int *) pcimage2;

   pbuf=pcimage2; for(i=0;i<total;i++)  *pbuf++=0;
   pbuf=pelem;  el_sum=0;
   for(i=0;i<elem_rows*elem_columns;i++)
        { j=(int) (*pbuf);  el_sum += j; pbuf++;  }
   el_sum = el_sum * value;
/*  printf("\n ival=%d   cval=%d  value=%d  el_sum=%d\n", ival, (int)cval,
               value, el_sum);
*/
   h_rows=elem_rows/2; h_col=elem_columns/2;
   pbi = (short int *) pimage;
   x=h_rows;
   do { y=h_col;
        do { sum=0;
             pbuf=pelem;
             for (i=0;i<elem_rows;i++)
                for (j=0;j<elem_columns;j++)
                   { el_temp=(int) (*pbuf); pbuf++;
                     if (n_bytes==2)
                             im_temp=(int) (*(pbi+(x-h_rows+i)*im_columns
                                                +y-h_col+j) );
                        else  im_temp=(int) (*(pimage+(x-h_rows+i)*im_columns
                                                +y-h_col+j) );
                     itemp = el_temp*im_temp;
                     sum += itemp;  }
/* if (sum) printf("\n %d   %d   %d   ", x,y,sum);  */
             if (sum == el_sum)
                   {    if (n_bytes==2)
                              *(pimage2+x*im_columns+y)=ival;
                        else
                              *(pcimage2+x*im_columns+y)=cval;
                    }
             y++;
            } while (y < im_columns-h_col);
        x++;
```

```
        }  while (x < im_rows-h_rows);

    pbuf=pimage;  pci=pcimage2;
    for(i=0;i<total;i++)  *pbuf++ = *pci++;
/*      {  if (*pci) printf("  %d (%d)  ", (int)*pci, i);
          *pbuf++ = *pci++;  }
*/
    free(pcimage2);
    return(value);

}


int morph_open(pimage, n_bytes, im_rows,im_col, pelem,elem_rows,elem_col)
int n_bytes, im_rows, im_col, elem_rows, elem_col;
 unsigned char *pimage, *pelem;
{  int i;

        i=erosion(pimage, n_bytes, im_rows,
                        im_col, pelem,elem_rows,elem_col);
    if (i>0)
        i=dilation(pimage, n_bytes, im_rows,
                        im_col, pelem,elem_rows,elem_col);
    return(i);
}



int morph_close(pimage, n_bytes, im_rows,im_col, pelem,elem_rows,elem_col)
int n_bytes, im_rows, im_col, elem_rows, elem_col;
unsigned char *pimage, *pelem;
{  int i;

    i=dilation(pimage, n_bytes, im_rows,im_col,
                        pelem,elem_rows,elem_col);
    if (i>0)
        i=erosion(pimage, n_bytes, im_rows,im_col,
                        pelem,elem_rows,elem_col);
    return(i);
}
```

APPENDIX D

C CODE FOR MR IMAGE SEGMENTATION

The segmentation algorithm is named "mri2". The user is prompted with a menu of segmentation options during analysis.

```
#include <stdio.h>
#include <fcntl.h>
#include <math.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/uio.h>
#include <malloc.h>

main(argc, argv)
int argc;
char *argv[];
  { int i,j;
   FILE *results;
   char subject_name1[25],subject_name2[25],output_file[25];
   char  *in_file_name,roi_file[30];
   short int *buf;
   int fd1, fd2;
    short  int image[256][256],out_image[256][256],out_image2[256][256];
    short in_image[256][256],total_area;
    short int temp_image[256][256],in_image2[256][256], image_temp[256][256];
    int msk[9],mm[5][5],holes[1000],hole_number,k;
    short corners[10],x1,x2,y1,y2;
    short int x_c,y_c,max_gray;
    short display_size,size=256,c,nd,std_ct;
    int mean_soft,soft,sum_soft_t,soft_t,cortical_t,out_of_plane,n_smooth,ends,contour_flag
   int contour_breaks,x_1,y_1,x_2,y_2,trabecular_bone;
   short sum_std,cortical;
   char *in,*out;
   double trab_t;
   float mean,std,trab_area,TBV;
   char reply;
    int x=1,y=1;
   if (argc<2) {  printf("TYPE: rdimg <file1> \n \n"); exit(0); }
   printf("\n");

   fd1=open(argv[1], 0);
if (fd1==-1) { printf("\n error \n"); exit(0); }

   fd2=open(argv[2], 1);

   for (i=0; i<256;i++)
        {
          read(fd1,   ( char*)image[i],512);
        }
        for (i=0;i<size;i++)
        for(j=0;j<size;j++)
        {
                in_image[i][j]=image[i][j];
        }


        in_file_name=argv[1];
/* extract filename and treat as ROI file */
        for (i=0; i<strlen(argv[1]); i++) {
                roi_file[i]=*(in_file_name+i);

                if (in_file_name[i]=='_') {
                        break;
                }
        }
        roi_file[i+1]='.';
        roi_file[i+2]='r';
        roi_file[i+3]='o';
        roi_file[i+4]='i';
        roi_file[i+5]='\0';
```

```
        /* open output file results.dat */
        printf("Enter name of output file to write results:> ");
        scanf("%24s",output_file);
        printf("Name of subject being analyzed:   \n");
        scanf("%s %s",subject_name1,subject_name2);
        if ((results=fopen(output_file,"w")) == (FILE *) NULL){
                printf("File Error");
                exit(1);
        }
        fprintf(results," RESULTS  FOR:  %s  %s\n",subject_name1,subject_name2);
do {
                printf("\n <---- BONE STRUCTURE ANALYSIS ----> \n");
                printf("\n          ( MAIN MENU ) \n\n");
                printf("        a. - Define ROI sub-image ... (# 10)\n");
                printf("        b. - Inverse Video ..... (# 11)\n");
                printf("        c. - Contour Bone ..... (# 12,19)\n");
                printf("        d. - Find trabecular area ... (# 13)\n");
                printf("        e. - Define principle axis .. (# 14)\n");
                printf("        f. - Adaptive threshold ..... (# 15) \n");
                printf("        g. - Perimeter analysis\n");
                printf("        h. - Skeleton by ZHANG-SUEN . (# 17)\n");
                printf("        i. - Indices of structure ... (# 20)\n");
                /*printf("          j. - Contour by distance transform\n");*/
                printf("        k. - Count holes in bone \n");
                printf("        l. - Count holes in bone with area > 1 pixel\n");
                printf("        m. - Strut orientation\n");
                printf("        q. - Quit\n");
                printf("\n What is your selection ? ");
        /*    reply=getchar(); */ scanf("%c", &reply);
              printf("\n");

              switch(reply) {
                      case 'a':
                              region_of_interest(image,out_image,corners,roi_file);
                              x1=corners[0];  y1=corners[1];
                              x2=corners[2]; y2=corners[3];
                              out="/MUMC/remote1/666_10.img";
                              write_image(out,out_image,256);
                              close (out);
                              break;
                      case 'b':

                              printf("maximum gray value= %d\n",maximum(out_image,size));

                              cortical_t=maximum(out_image,size);
                              for (i=y1;i<=y2;i++){
                                      for (j=x1;j<=x2;j++){
                                      }
                              }
                              cortical_t=0.2*maximum(out_image,size);
                              if (cortical_t>250){
                                      cortical_t=250;
                              }
                              out="/MUMC/remote1/666_11.img";
                              write_image(out,out_image,256);
                              close(out);
                              break;
                      case 'c':
                              in="/MUMC/remote1/666_10.img";
                              read_image(in,out_image,256);
                              cortical_t=maximum(out_image,size);
                              optimize_contrast(out_image,256,x1,x2,y1,y2);
                              for (i=1; i<=9; i++)
                                      msk[i]=1;
                             printf(" How many times to smooth image ?\n");
```

```
scanf("%d",&n_smooth);
printf("Smoothing .....10X\n");
/* n_smooth=5;  */
for (i=1; i<=n_smooth; i++) {
        convolve_mask(out_image,msk,9,x1,x2,y1,y2,256);
}
msk[1]=0; msk[2]=-1; msk[3]=0;
  msk[4]=-1; msk[5]=8; msk[6]=-1;
  msk[7]=0; msk[8]=-1; msk[9]=0;
convolve_mask(out_image,msk,8,x1,x2,y1,y2);
/*optimize_contrast(out_image,256,x1,x2,y1,y2,corners[2]);*
for (i=0;i<size;i++)
        for (j=0;j<size;j++) {
                temp_image[i][j]=out_image[i][j];
                 out_image[i][j]=0;
         }
out="/MUMC/remote1/666_19.img";
write_image(out,temp_image,256);
printf("Input Cortical and subcortical= %d\n",cortical_t);
scanf("%d",&cortical_t);

contour(temp_image,out_image,size,x1,x2,y1,y2,cortical_t);


for (i=y1;i<=y2;i++)
        for (j=x1;j<=x2;j++) {
                temp_image[y1][j]=255; temp_image[y2][j]=25
                temp_image[i][x1]=100; temp_image[i][x2]=10
         }
/* why this ?? */
out="/MUMC/remote1/666_19.img";
write_image(out,temp_image,256);
close(out);
 y=(y1+y2)/2;
 for (x=x1; x<=x2; x++){
        if(temp_image[y][x]==255)
                 break;
 }
total_area=grow_region(temp_image,256,256,y,x,250);
/*printf("\n area=%d\n", total_area);*/
out="/MUMC/remote1/666_12.img";
write_image(out,temp_image,256);
close(out);
c=1;
while (total_area <500){  /* check that area grown is big e
        in="/MUMC/remote1/666_12.img";
        read_image(in,temp_image,256);
        total_area=grow_region(temp_image,256,256,y,x+c,250
        c++;
}
for (i=y1;i<=y2;i++)
        for (j=x1;j<=x2;j++) {
                if(temp_image[i][j]==3000)
         }
out="/MUMC/remote1/666_12.img";
write_image(out,temp_image,256);
close(out);
printf("  \n");
printf("          ****************************************
printf("          * OPEN DISPLAY AND VERIFY THAT CONTOUR IS
printf("          ****************************************
printf("  \n");
printf("Is contour broken  ? (1=YES  0=NO)\n");
scanf("%d",&contour_flag);
if(contour_flag==0) continue; {
        printf("How many sets of breaks to close ?\n");
        scanf("%d",&contour_breaks);
```

```
                for (k=1; k<= contour_breaks; k++){
                        printf("Input co-ordinates of end #1\n");
                        scanf("%d %d",&y_1,&x_1);   /* ordinates a
                        printf("Input co-ordinates of end #2\n");
                        scanf("%d %d",&y_2,&x_2);
                        printf("Closing ..... break # %d\n",k);
                        close_contour(temp_image,size,x_1,x_2,y_1,
                }
        }
        for (i=0;i<size;i++)
                for (j=0;j<size;j++) {
                        out_image[i][j]=temp_image[i][j];
                }
        distance_transform(out_image,256); /* highlight border to s
        for (i=0;i<size;i++)
                for (j=0;j<size;j++) {
                        if(out_image[i][j]==2) {
                                temp_image[i][j]=100;
                        }
                }

        /* label border of image with value 100 */
        for (i=y1;i<=y2;i++)
                for (j=x1;j<=x2;j++) {
                        temp_image[y1][j]=100; temp_image[y2][j]=10
                        temp_image[i][x1]=100; temp_image[i][x2]=10
                }
        out="/MUMC/remote1/666_12.img";
        write_image(out,temp_image,256);
        close(out);
        break;

case 'd':
        in="/MUMC/remote1/666_12.img";
        read_image(in,out_image,256);
        x=(x1+x2)/2; y=(y1+y2)/2;
        printf("   \n");
        trab_area=(float) fill_contour(out_image,256,256,y,x,100)*(
        printf("\n Trabecular Area=%.2f mm2\n", trab_area);
        fprintf(results,"Trabecular Bone Area=%.2f mm2\n", trab_are
        in="/MUMC/remote1/666_11.img";
        read_image(in,in_image2,256);
        for (i=0;i<size;i++)
                for(j=0;j<size;j++) {
                        if(out_image[i][j]==3000){
                                image_temp[i][j]=255;
                        }
                        else {
                                out_image[i][j]=0;
                                image_temp[i][j]=0;
                        }

                }
        out="/MUMC/remote1/666_13.img";
        write_image(out,out_image,256);
        close(out);
        distance_transform(image_temp,256); /* define inner contour
        for (i=0;i<size;i++)
                for (j=0;j<size;j++) {
                        temp_image[i][j]=0;
                        if(image_temp[i][j]==2) {
                                temp_image[i][j]=255;
                        }
                }
        out="/MUMC/remote1/666_19.img";
        write_image(out,temp_image,256);
```

```
                close(out);
                break;

        case 'e':
                for (i=0;i<size;i++)
                        for (j=0;j<size;j++) {
                                out_image[i][j]=0;
                        }
                in="/MUMC/remote1/666_19.img"; /* read in contour image */
                read_image(in,in_image2,256);

                moment_of_inertia(in_image2,out_image,size,x1,x2,y1,y2,mm);
                x_c=mm[1][0]/mm[0][0]; y_c=mm[0][1]/mm[0][0];
                printf("centroid = %d  %d\n",x_c,y_c);
                out="/MUMC/remote1/666_14.img";
                write_image(out,out_image,256);
                close(out);
                break;

        case 'f':
                in="/MUMC/remote1/666_13.img"; /* read in filled contour im
                read_image(in,in_image2,256);
                in="/MUMC/remote1/666_13.img";
                read_image(in,out_image,256);
                optimize_contrast(in_image2,256,x1,x2,y1,y2);
                optimize_contrast(out_image,256,x1,x2,y1,y2);
                trabecular_bone=adaptive_t(in_image2,out_image,256,4); /* 4
                TBV=(((float)trabecular_bone*0.195*0.195)/trab_area)*100.0;
                printf("\n Trabecular Bone Volume (TBV)=%.2f %% \n", TBV);
                fprintf(results,"Trabecular Bone Volume (TBV)=%.2f %%\n", T
                in="/MUMC/remote1/666_19.img";
                read_image(in,in_image2,256);
                add_images(out_image,in_image2,size); /* add contour to bin
/* for (i=y1;i<=y2;i++)
  for (j=x1;j<=x2;j++) {
        if((i<y1+4)| (i>y2-4)){
                out_image[i][j]=0;
        }
 } */

                out="/MUMC/remote1/666_15.img";
                write_image(out,out_image,256);
                close(out);
                break;

        case 'g':
                 /* Perimeter analysis */
                in="/MUMC/remote1/666_15.img";
                read_image(in,out_image,256);
                distance_transform(out_image,256); /* define inner contour
                for (i=0;i<size;i++)
                        for (j=0;j<size;j++) {
                                if (out_image[i][j]==2){
                                        temp_image[i][j]=255;
                                }
                                else {
                                        temp_image[i][j]=0;
                                }
                   }

                out="/MUMC/remote1/666_18.img";
                write_image(out,temp_image,256);
                close(out);
                /* skeleton out of plane-connected bone structure
                in="/MUMC/remote1/666_16.img";
                read_image(in,out_image2,256);
```

```
                for (i=0;i<size;i++)
                        for (j=0;j<size;j++) {
                                temp_image[i][j]=out_image2[i][j];
                                 out_image2[i][j]=0;
                         }
                out="/MUMC/remote1/666_18.img";
                write_image(out,out_image2,256);
                close(out);
                in="/MUMC/remote1/666_17.img";
                read_image(in,in_image2,256);
                in="/MUMC/remote1/666_18.img";
                read_image(in,out_image,256);
                add_images(out_image,in_image2,size);
                out="/MUMC/remote1/666_19.img";
                write_image(out,out_image,256);*/
                break;



        case 'h':
                /* skeleton in plane-connected bone structure  by Zhang-Sue
                in="/MUMC/remote1/666_15.img";
                read_image(in,out_image,256);

                thinzs(out_image,255);
                out="/MUMC/remote1/666_17.img";
                write_image(out,out_image,256);
                close(out);
                /* skeleton out of plane bone structure  by Zhang-Suen algo
                in="/MUMC/remote1/666_16.img";
                read_image(in,out_image,256);

                thinzs(out_image,255);
                out="/MUMC/remote1/666_18.img";
                write_image(out,out_image,256);
                close(out);
                break;
        case 'i':
                /* Calculate indices for in-plane structure */
                in="/MUMC/remote1/666_17.img";
                read_image(in,out_image,256);
                for (i=0;i<size;i++)
                        for (j=0;j<size;j++) {
                                temp_image[i][j]=out_image[i][j];
                                out_image[i][j]=0;
                        }
                printf(" *** IN PLANE STRUCTURE ***\n");
                indices(temp_image,out_image,256,256,255,500,1000,results);
                out="/MUMC/remote1/666_20.img";
                write_image(out,out_image,256);
                close(out);
                break;
        case 'j':
                in="/MUMC/remote1/666_12.img";
                read_image(in,out_image,256);
                distance_transform(out_image,256);
                /* local_maximum(out_image,temp_image); */
                for (i=0;i<size;i++)
                   for (j=0;j<size;j++) {
                        temp_image[i][j]=0;
                   }
                for (i=0;i<size;i++)
                   for (j=0;j<size;j++) {
                        if(out_image[i][j]==2) {
                                temp_image[i][j]=255;}
```

```
                }
                out="/MUMC/remote1/666_14.img";
                write_image(out,temp_image,256);

                in="/MUMC/remote1/666_14.img";
                read_image(in,in_image2,256);
                in="/MUMC/remote1/666_15.img";
                read_image(in,out_image,256);
                add_images(out_image,in_image2,size);
                out="/MUMC/remote1/666_15.img";
                write_image(out,out_image,256);
                close(out);
                break;
        case 'k':
                in="/MUMC/remote1/666_15.img";
                read_image(in,out_image,256);


                /* label border of image with value=200 */
                for (i=y1;i<=y2;i++)
                        for (j=x1;j<=x2;j++) {
                                out_image[i][j]=out_image[i][j]*4;
                                temp_image[i][j]=out_image[i][j];
                                temp_image[y1][j]=2000; temp_image[y2][j]=2
                                temp_image[i][x1]=2000; temp_image[i][x2]=2
                        }
                hole_number=hole_counter(holes,temp_image,x1,x2,y1,y2)-1;
                printf(" holes detected= %d\n",hole_number);

                /* fill in all holes with area equal to 1 pixel */

                in="/MUMC/remote1/666_15.img";
                read_image(in,out_image,256);
                for (k=0; k<=hole_number; k++) {
                        if (holes[k]==1) {
                                for (i=y1;i<= y2;i++) {
                                        for (j=x1;j<= x2;j++) {
                                                if (temp_image[i][j]== k+1)
                                                        temp_image[i][j]=25
                                                        out_image[i][j]=255
                                                }
                                        }
                                }
                        }
                }

                hole_distribution(holes,hole_number,2);  /* determine hole
                mean_std(holes,mean,std,hole_number,0.195,results); /* pixe
                sort(holes,hole_number,0.195,results);
                out="/MUMC/remote1/666_16.img";
                write_image(out,out_image,256);
                close(out);
                break;

        case 'l':
                for (i=0; i<1000; i++) /* clear holes */
                        holes[i]=0;

                in="/MUMC/remote1/666_16.img";
                read_image(in,out_image,256);
                /* label border of image with value=200 */
                for (i=y1;i<=y2;i++)
                        for (j=x1;j<=x2;j++) {
                                out_image[i][j]=out_image[i][j]*4;
                                temp_image[i][j]=out_image[i][j];
                                temp_image[y1][j]=2000; temp_image[y2][j]=2
```

```
                                                temp_image[i][x1]=2000; temp_image[i][x2]=
                                        }

                                        hole_number=hole_counter(holes,temp_image,x1,x2,y1,y2)-1;
                                        printf(" holes with area > 1 pixel detected= %d\n",hole_nu
                                        fprintf(results,"    \n");
                                        fprintf(results,"-----------------------------------------
                                        fprintf(results,"  Hole size analysis for holes > 1 pixel
                                        fprintf(results,"-----------------------------------------
                                        /*hole_distribution(holes,hole_number,2);   */
                                        mean_std(holes,mean,std,hole_number,0.195,results);
                                        sort(holes,hole_number,0.195,results);
                                        break;
                                case 'm':
                                        in="/MUMC/remote1/666_13.img";
                                        read_image(in,out_image,256);
                                        orientation(out_image,temp_image,size,x1,x2,y1,y2,results);
                                        break;

                                        }
        } while (reply != 'q');

/*  Write to MUMC display file */
    for (i=0; i<256;i++) {
            write(fd2, (char*)out_image[i],512);
    }

    close(fd1);
    close(fd2);
    exit(0);
}

/********** Function to read a MUMC display image *************/

int read_image(in_file,values,m_size)
char *in_file;
short m_size,values[256][256];
{
short int fp,i;
fp=open(in_file,0);
for (i=0; i<m_size;i++)
        {
        read(fp, (char*)values[i],m_size*2);
        }
 close(fp);
 }


/********** Function to write image to MUMC display *************/

int write_image(out_file,values,m_size)
char *out_file;
short values[256][256],m_size;
{
int fp,i;
fp=open(out_file,1);
for (i=0; i<m_size;i++)
        {
        write(fp, (char*)values[i],m_size*2);
        }

   close(fp);
}


/********** Function to convolve image with a mask ***************/
```

```
int convolve_mask(values,m,sf,rx1,rx2,ry1,ry2,m_size)
 short int values[256][256],rx1,rx2,ry1,ry2,sf,m_size;
int m[9];
{  int i,j,k;

     short int in_values[256][256];
     short int values_x[256][256];
         int x=1,h=1,y=1;
         int m1,m2,m3,m4,m5,m6,m7,m8,m9;
         int sum1,sum2,sum3;
         double sum_mask;

if((ry1-10) < 0)
         ry1=1;
if((ry2+10)>m_size)
         ry2=m_size-1;

   for (i=ry1;i<= ry2;i++)
         for(j=rx1;j<= rx2;j++) {
                 in_values[i][j]=values[i][j];
         }

         x=ry1;
         do {
         y=rx1;
         do{
sum1=(m[1]*values [x-1][y-1])+(m[2]*values[x-1][y])+(m[3]*values[x-1][y+1]);     sum2=(m[4]*
sum3=(m[7]*values[x+1][y-1])+(m[8]*values[x+1][y])+(m[9]*values[x+1][y+1]);
         values_x[x][y]=(sum1+sum2+sum3)/sf;
        if (values_x[x][y] > 32000 )
                 values_x[x][y]=0;
         if (values_x[x][y] < 0){
                 values_x[x][y]=0;
         }
         y=y+1;
         }
         while (y <=(rx2-1));
         x=x+1;
         }
         while (x<=(ry2-1));

         for (i=ry1; i<=ry2; i++)
         for (j=rx1; j<=rx2; j++){
                 values[i][j]=values_x[i][j];
                 if (values[i][j] <0)
                         values[i][j]=0;
                 }
}


/********** Function to apply median filter to image *****************/
int median_filter(values,in_values,m_size)
short int values[256][256],m_size,in_values[256][256];
{  int i,j,k;
    short int out_values[256][256],m[9];
         int x=1,h=1,y=1,lp,size;
         int row, column, num_median;
         int temp;

         row=m_size;
         column=m_size;
/* *** Apply median filter  *** */
         x=2;
         do
         {
         y=2;
```

```
        do
        {
         m[0]=values [x-1][y-1];
         m[1]=values[x-1][y];
         m[2]=values[x-1][y+1];
         m[3]=values[x][y-1];
         m[4]=values[x][y];
         m[5]=values[x][y+1];
         m[6]=values[x+1][y-1];
         m[7]=values[x+1][y];
         m[8]=values[x+1][y+1];
                for (i=0; i < 8; ++i)
                {
                        for(j=i+1; j < 9; ++j)
                        {
                                if (m[i] > m[j])
                                {
                                        temp=m[i];
                                        m[i]=m[j];
                                        m[j]=temp;
                                }
                        }
                }
        out_values[x][y]=m[5];
        y=y+1;
        }
        while (y <=(column-1));
        x=x+1;
        }
        while (x<=(row-1));

        for (i=0;i<m_size;i++)
        for(j=0;j<m_size;j++)
        {
         values[i][j]=out_values[i][j];
        }

 }

int add_images (values,values2,m_size)
short int values[256][256],values2[256][256],m_size;
{
        int max_value,i,j;
        max_value=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                values[i][j]=values[i][j]+values2[i][j];
                if (values[i][j] > 255)
                {values[i][j]=255;}
         }
}

int subtract_images (values,values2,m_size)
short int values[256][256],values2[256][256],m_size;
{
        int max_value,i,j;
        max_value=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                values[i][j]=values[i][j]-values2[i][j];
                if (values[i][j] < 0)
                {values[i][j]=0;}
         }
}
```

```
/********** Function to determine corners of rectangular roi **********/
int region_of_interest (temp_in,temp_out,ct,roi_f)
short ct[100],temp_in[256][256],temp_out[256][256];
 char roi_f[30];
{
        short *buf2;
        short *fpoint;
        int handle, bytes,i,j;
        short scale;
        short display_size;
        buf2=(short *) calloc(300,1);

        printf("Reading ROI file .....");
        if ((handle=open(roi_f,O_RDONLY ))==-1){
                printf("Error opening file.\n");
                exit(1);
        }
        if((bytes=read(handle,buf2,200))==-1){
                printf("Read failed");
                exit(1);
        }
         else {
                printf("Read: %d bytes read.",bytes);
        }

/* ** Get information from MUMC ROI file ** */

        fpoint=buf2;
        for (i=0; i< 9; i++)
                {fpoint++;}
        display_size=*fpoint;
        fpoint++;
        for (i=0; i< 30; i++){
                ct[i]= *fpoint;
                fpoint++;
        }
        close(handle);
        scale=display_size/256;
        ct[0]=ct[0]/scale;
        ct[1]=(512-ct[1])/scale;
        ct[2]=ct[2]/scale;
        ct[3]=(512-ct[3])/scale;

/* Set outside roi to 0 */
        for(i=ct[1]; i<=ct[3]; i++)
                for(j=ct[0]; j<=ct[2]; j++){
                        temp_out[i][j]=temp_in[i][j];
                }
free(buf2);
}

/************ Function to find histogram of image matrix ***************/

int histogram(values,m_size,bin_size)
short int values[256][256],m_size,bin_size;
{
  int i,j,bin,max_ct;
  short int histo[256],h[256],slope[256],max[20],ct=0;

        for (i=0;i<m_size;i++)
        for(j=0;j<m_size;j++)
        {
                if(values[i][j] !=0)
                {bin=values[i][j]/bin_size;
```

```
                histo[bin]=histo[bin]+1; }
        }
 /* Apply a 5-point smooth to histogram */
        for(j=0;j<m_size;j++)
        {
      h[j]=(histo[j]+2*histo[j+1]+3*histo[j+2]+2*histo[j+3]+histo[j+4])/9;
        }
        for(i=4; i<m_size; i++)
                {slope[i]=h[i+1]-h[i];}
        for(i=4; i<256; i++)
        {
        if(slope[i]>0 && slope[i+1]<0)
                {max[ct]=i+2;
                ct++;}
        }
        max_ct=max[0];
        return(max_ct);
 /*printf("max= %d\n",max[0]);*/
 }

 /************ Function to find the maximum value of an array ************/

 int maximum (values,m_size)
 short int values[256][256],m_size;
 {
        int max_value,i,j;
        max_value=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
        {
                if (values[i][j] > max_value)
                { max_value=values[i][j];}
        }
        return(max_value);
 }

 /****** Function to optimize contrast using histogram equalization *****/
 int optimize_contrast(values,m_size,rx1,rx2,ry1,ry2)
 short int values[256][256],m_size,rx1,rx2,ry1,ry2;
 {
 int hist[12500];
 int npix,max_pixel,i,j;
 for (i=0;i< 12500;i++)
        hist[i]=0;

 for (i=ry1;i<=ry2;i++)
        for(j=rx1;j<=rx2;j++)
        {
                hist[values[i][j]]++;
        }
        /*npix = m_size*(m_size-1) - hist[0];*/
        npix= ((rx2-rx1)*(ry2-ry1))-hist[0];
        hist[0] = 0;

max_pixel=maximum(values,m_size);
printf("\n Maximum ct#=%d\n", max_pixel);
printf("\n npix= %d\n",npix);
        for (i = 1; i < max_pixel; i++)
                hist[i] += hist[i-1];

        for (i=0;i<m_size;i++)
        for(j=0;j<m_size;j++)
                {
                values[i][j] = (hist[values[i][j]] * max_pixel/npix);
                }
}
```

```
/************ Function to find the mean value of an array *************/

int mean (values,m_size)
short int values[256][256],m_size;
{
        int mean_value,sum,mean_counter,i,j;
        sum=0;
        mean_counter=0;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                if (values[i][j] !=0)
                { sum=values[i][j]+sum;
                 mean_counter++;}
         }
        mean_value=sum/mean_counter;
        return(mean_value);
}

/******************** Function to threshold an image ***************/
/* Values above threshold remain the same, below T set to 0 */

int thresholding (values,values2,m_size,T)
short int values[256][256],values2[256][256],m_size,T;
{
        int i,j,counter;
        for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                if ((values2[i][j] > 1300) || (values[i][j] > T ))
                        {values[i][j]=255;
                        counter++;}
                else {
                        values[i][j]=0;}
        }
}


/************ Function to determine the moments of image *************/
int moment_of_inertia (values,temp_image,m_size,rx1,rx2,ry1,ry2,mom)
short int values[256][256],temp_image[256][256],m_size,rx1,rx2,ry1,ry2;
 int mom[5][5];
{
        int mean_value,total_sum,x,y;
        int i,j,f,x_m,y_m,b_term,x_mean,y_mean;
        int sum00,sum01,sum10,sum11,sum20,sum02;
        double sum_mask,tan_theta1,tan_theta2,f_dec;
        sum00=0;
        sum10=0; sum01=0;
        for (x=0; x< 6; x++)
                for (y=0; y< 6; y++)
                        {mom[x][y]=0;}

        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                if (values[x][y] >0)
                        {sum00=values[x][y]+sum00;
                        sum01=y*values[x][y]+sum01;
                        sum10=x*values[x][y]+sum10;
                        }
         }
        mom[0][0]=sum00;
        mom[0][1]=sum01;
        mom[1][0]=sum10;
        x_m=mom[1][0]/mom[0][0]; y_m=mom[0][1]/mom[0][0];
```

```
 /* Determine principle axis... least moment of inertia */

        sum11=0;
        sum20=0; sum02=0;
        for (x=0; x< m_size; x++)
                for (y=0; y< m_size; y++){
                if (values[x][y] >0){
                        sum11=(x-x_m)*(y-y_m)*values[x][y]+sum11;
                        sum20=(x-x_m)*(x-x_m)*values[x][y]+sum20;
                        sum02=(y-y_m)*(y-y_m)*values[x][y]+sum02;
                }
         }
        mom[1][1]=sum11;
        mom[2][0]=sum20;
        mom[0][2]=sum02;

                x_mean=mom[1][0]/mom[0][0];
                y_mean=mom[0][1]/mom[0][0];
                printf("Central moments (x,y)=%d  %d\n",x_mean,y_mean);
                b_term=(mom[2][0]-mom[0][2])/mom[1][1];
                sum_mask=(b_term*b_term)+4;
                tan_theta1=0.5*(-1*b_term+sqrt(sum_mask));
                tan_theta2=0.5*(-1*b_term-sqrt(sum_mask));
                printf("tan_theta1= %f\n",tan_theta1);
                printf("tan_theta2= %f\n",tan_theta2 );
temp_image[x_mean][y_mean]=2500;

/* Define principle axis */
        for(i=ry1; i<=ry2; i++)
        for (f=0; f<10; f++)
                {
                        f_dec=0.1*f;
                        j=tan_theta1*((i+f_dec)-x_mean)+y_mean;
                        if((j>=rx1-10)&&(j<=rx2+10))
                                {temp_image[i][j]=2500;}
                        j=tan_theta2*((i+f_dec)-x_mean)+y_mean;
                if((j>=rx1-10)&&(j<=rx2+10))
                        {temp_image[i][j]=2500;}
                }
}

/************ Function to check trabecular orientation *************/
int orientation (values,temp_image,m_size,rx1,rx2,ry1,ry2,res_file)
short int values[256][256],temp_image[256][256],m_size,rx1,rx2,ry1,ry2;
FILE *res_file;
{
        int i,j,bin,sum;
        short int a[9],g_x,g_y,histo[40];
        double sum_mask,theta,G,tan_theta,temp_theta;
        char *buf2,*fpoint2;
        char out_file[25];
        FILE *out;

/* clear histogram */
        for (i=0; i <=40; i++)
                histo[i]=0;


/* Define gradient from Sobel mask */
        for(i=ry1; i<=ry2; i++)
                for (j=rx1; j<=rx2; j++){
                        if(values[i][j] >0) {
                                a[0]=values[i-1][j-1];
                                a[1]=values[i-1][j];
                                a[2]=values[i-1][j+1];
```

```
                                        a[3]=values[i][j+1];
                                        a[4]=values[i+1][j+1];
                                        a[5]=values[i+1][j];
                                        a[6]=values[i+1][j-1];
                                        a[7]=values[i][j-1];
                                        g_x=(a[2]+2*a[3]+a[4])-(a[0]+2*a[7]+a[6]);
                                        g_y=(a[0]+2*a[1]+a[2])-(a[6]+2*a[5]+a[4]);
                                        sum_mask=(g_x*g_x)+(g_y*g_y);
                                        if (sum_mask==0) continue;
                                        G=sqrt(sum_mask);
                                        if (G<1500) continue;
                                        /* temp_image[i][j]=G;*/
                                        tan_theta=(double)g_x/(double)g_y;
                                        theta=(atan(tan_theta)/3.14159)*180;
                                        if(theta <0 ){
                                                theta=180+theta;
                                        }
                                        bin=theta/5;
                                        histo[bin]=histo[bin]+1;
                                }
                        }

 /* *** write to file in ascii format *** */
                buf2=(char*) malloc(5000);
                printf("Enter output file for trabecular orientation:> ");
                scanf("%24s",out_file);
                if ((out=fopen(out_file,"w+t")) == (FILE *) NULL){
                        printf("File Error");
                        exit(1);
                }
                fprintf(res_file,"Orientation File= %s\n",out_file);
                for(i=0; i<=35; i++){
                        sum +=histo[i];
                }

                fpoint2=(char*) buf2;
                fprintf(out,"   \n");
                fprintf(out,"   Degrees     Orientation Factor \n");
                for(i=0; i<=35; ++i){
                        sprintf(fpoint2, "%5d                %.3f\n",(i*5)+5,((float)histo[i]/sum)
                        fputs (fpoint2, out);
                }
free(buf2);
}

/********** Function to remove isolated points  ***********/
int clean_image (values,m_size,p)
short int values[256][256],m_size,p;
{
        int i,j,m[9];
        for (i=2; i< m_size-1; i++)
        for (j=2;j< m_size-1; j++)
         {
         m[0]=values[i-1][j-1];
         m[1]=values[i-1][j];
         m[2]=values[i-1][j+1];
         m[3]=values[i][j-1];
         m[4]=values[i][j];
         m[5]=values[i][j+1];
         m[6]=values[i+1][j-1];
         m[7]=values[i+1][j];
         m[8]=values[i+1][j+1];
if((m[4]==p)&&(m[0]==p)&&(m[1]==p)&&(m[2]==p)&&(m[3]==0)&&(m[5]==0)&&(m[6]==0)&&(m[7]==0)&&
                        {values[i][j]=0;}
if((m[4]==p)&&(m[0]==0)&&(m[1]==0)&&(m[2]==0)&&(m[3]==0)&&(m[5]==0)&&(m[6]==0)&&(m[7]==0)&&
                        {values[i][j]=0;}
```

```
   if((m[4]==0)&&(m[0]==p)&&(m[1]==p)&&(m[2]==p)&&(m[3]==p)&&(m[5]==p)&&(m[6]==p)&&(m[7]==p)&
                              {values[i][j]=p;}

          }
   }

   /****************** Function to grow region given a seed pixel *************/
   int stack=0;
   short int *pstack;

   int grow_region(pimage,im_rows,im_columns,xc,yc,t)
   short *pimage;
   int im_rows,im_columns;
   int xc,yc;
   short t;
   {
      int i,j,x,y;
       short *pbuf;

      pstack = (short int *) malloc(32000);
      if (pstack==NULL)
           { printf("\nGrow region: Not enough memory\n");  return(-1); }

      px_push(xc,yc);
      *(pimage+(xc*im_columns)+yc)=3000;

      while(stack)
           {
             px_pop(&x,&y);

            i=(x-1)*im_columns; j=y;
             if (*(pimage+i+j)==3000) i=i;
               else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                            px_push(x-1,y);      }
                    else *(pimage+i+j)=0;

             i=(x+1)*im_columns; j=y;
             if (*(pimage+i+j)==3000) i=i;
               else if (*(pimage+i+j)>=t) {*(pimage+i+j)=3000;
                                            px_push(x+1,y);      }
                    else *(pimage+i+j)=0;

             i=x*im_columns; j=y-1;
             if (*(pimage+i+j)==3000) i=i;
               else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                            px_push(x,y-1);      }
                    else *(pimage+i+j)=0;

             i=x*im_columns; j=y+1;
             if (*(pimage+i+j)==3000) i=i;
               else if (*(pimage+i+j)>=t) { *(pimage+i+j)=3000;
                                            px_push(x,y+1);      }
                    else *(pimage+i+j)=0;
           }


      x=0;
      for (i=0;i<im_rows*im_columns;i++)
           {      if (*(pimage+i)!=3000) *(pimage+i)=0;
                  else  x++;     }
     free(pstack);
     return(x);
   }

   int px_push(x,y)
   int x;
```

```
  int y;
  {
    *pstack= (short int)x;
    pstack++;
    *pstack= (short int)y;
    pstack++;
    stack++;
  /* printf("\n %d  (%d,%d)",stack, x,y);*/
  }

  int px_pop(px, py)
  int *px;
  int *py;
  {
    short int i;
    pstack--;
    i=*pstack;
    *py=(int)i; pstack--;
    i=*pstack;
    *px=(int)i;
    stack--;
  }

  /*************** Function to close bone contour  *****************/
  int close_contour(values,m_size,rx1,rx2,ry1,ry2)
  short int values[256][256],m_size,rx1,rx2,ry1,ry2;
  {
          int i,j,n=1,f;
          int delta,run_length,threshold,x[3],y[3];
          double slope,f_dec;


          x[1]=rx1; x[2]=rx2;
          y[1]=ry1; y[2]=ry2;

  /* define line between the end points */
          slope= ((double) (y[2]-y[1]))/((double)(x[2]-x[1]));
          printf("slope = %f\n",slope);
  /* draw line between ends */
          for (i=x[1]; i<=x[2]; i++) {
                  for (f=0; f<10; f++) {
                          f_dec=0.1*f;
                          j=slope*(i+f_dec-x[1])+y[1];
                          values[i][j]=255;
  /* make this missing bit of contour 3 pixel widths thick */
                          values[i][j-1]=255;
                          values[i][j+1]=255;
                          values[i-1][j]=255;
                          values[i+1][j]=255;
                  }
          }
  }
  /*************** Function to define extract cortical bone  *****************/
  int get_cortex(values,temp,m_size,T)
  short int values[256][256],temp[256][256],m_size,T;
  {
          int i,j,p1,p2,p3,p4;

  /* clear temp */
          for (i=0; i< m_size; i++)
          for (j=0;j< m_size; j++)
           {
                  temp[i][j] =0;
           }

  i=2;
```

```
 do
 {

        for(j=2; j<=m_size-1; j++)
        {
        p1=values[i][j];
        p2=values[i][j+1];
        p3=values[i-1][j+1];
        p4=values[i][j-1];

         if ((p1> T)&&(p2 > T || p3>T || p4>T))
                {
                 temp[i][j]=1;
                }
        }
 i++;
 }
 while(i<=m_size-1);
 }

 /*************** Function to define bone contour *****************/
 int contour(values,contour,m_size,rx1,rx2,ry1,ry2,T)
 short int values[256][256],contour[256][256],m_size,rx1,rx2,ry1,ry2,T;
 {
        int i,j,p1,p2,p3,p4,start_pixel=0;
        int delta,run_length,threshold;
        int temp[256][256];

 /* clear temp */
        for (i=0; i< m_size; i++){
                for (j=0;j< m_size; j++){
                        temp[i][j] =0;
                }

        }
 printf("Getting start point for contour\n");
 printf("Extracting cortex .....\n");

 /*for(j=rx1; j<=rx2; j++){
        values[ry1][j]=500;
        values[ry2][j]=500;
 }*/

         for (i=ry1; i<=ry2; i++){
                for(j=rx1; j<=rx2; j++){
                        p1=values[i][j];
                        p2=values[i][j+1];
                        p3=values[i-1][j+1];
                        p4=values[i][j-1];
                        if ((p1<T)&&(p2< T || p3<T || p4<T)){
                                        temp[i][j]=255;
                        }
                }
        }


 /* save cortical shell **/
 for (i=0; i< m_size; i++)
        for (j=0;j< m_size; j++)
         {
                values[i][j]=temp[i][j];
         }

 goto jump;
```

```
  /* define contour one pixel thick
   Scan along a row */
  threshold=99;
  i=ry1+2;
  do
  {

          for(j=rx1+2; j<rx2-2; j++)
          {
          run_length=0.5*(rx1+rx2);
          if (j>run_length)
          {goto next_i;}

          delta=temp[i][j]-temp[i][j-1];
           if ((abs(delta)> threshold) )
                  {
                  contour[i][j-1]=255;
                  goto next_i;
                  }
          }
 next_i:
 i++;
 }
 while(i<ry2-2);

 /* Scan down a column */
 j=rx1+2;
 do
 {
          for(i=ry1+2;i<ry2-2;i++)
          {
          run_length=0.5*(ry1+ry2);
          if (i>run_length)
          {goto next_j;}

          delta=temp[i][j]-temp[i-1][j];
           if ((abs(delta)> threshold))
                  {
                   contour[i-1][j]=255;
                  goto next_j;}
          }
 next_j:
 j++;
 }
 while(j<rx2-2);



 /* Scan back along a row */
 i=ry1+2;
 do
 {

          for(j=rx2-2; j>rx1+2; j--)
          {
          run_length=0.5*(rx1+rx2);
          if (j<run_length)
          {goto next_x;}
          delta=temp[i][j]-temp[i][j-1];
           if ((abs(delta)> threshold))
                  {
                   contour[i][j]=255;
                  goto next_x;}
          }
next_x:
```

```
 i++;
 }
 while(i<ry2-2);

  /*Scan back along a column */
 j=rx1+2;
 do
 {
        for(i=ry2-2; i>ry1+2; i--)
        {
        run_length=0.5*(ry1+ry2);
        if (i<run_length)
        {goto next_y;}
        delta=temp[i][j]-temp[i-1][j];
         if ((abs(delta)> threshold))
                {
                contour[i][j]=255;
                goto next_y;}
        }
next_y:
j++;
}
while(j<rx2-2);
jump:
        printf("end of contour");

}

/************** Function to check that contour is closed  *************/
int contour_check(in_values,values,rows,columns,val,fe_label)
short values[256][256],in_values[256][256];
int rows,columns,fe_label,val;
{
int x,y,node_sum,node=0,free_ends=0;
int i,j,isolated_points=0,network_length=0,m[9];
int k,count;

for(i=0; i<=9; i++)
        m[i]=0;      /* clear m to 0 */

/* Score for free ends */
node_sum=0;
for (x=2; x<=(rows-1); x++)
        for (y=2; y <= (columns-1); y++) {
                values[x][y]=0;
        }

for (x=2; x<=(rows-1); x++)
        for (y=2; y <= (columns-1); y++) {
                m[0]=in_values [x-1][y-1];
                m[1]=in_values[x-1][y];
                m[2]=in_values[x-1][y+1];
                m[3]=in_values[x][y-1];
                m[4]=in_values[x][y];
                m[5]=in_values[x][y+1];
                m[6]=in_values[x+1][y-1];
                m[7]=in_values[x+1][y];
                m[8]=in_values[x+1][y+1];
                node_sum=m[0]+m[1]+m[2]+m[3]+m[5]+m[6]+m[7]+m[8];
                        if((m[4]>0)&&(node_sum==val)){
                                values[x][y]=fe_label; /* label free ends  */
                                free_ends++;
                        }
        }
 return(free_ends);
```

```
}


/************* Function to  fill in contour given a seed pixel **************/
int stack2=0;
short int *pstack2;

int fill_contour(pimage2,im_rows,im_columns,xc,yc,t)
short *pimage2;
int im_rows,im_columns;
int xc,yc;
short t;
{
   int i,j,x,y;
    short *pbuf;
   pstack2 = (short int *) malloc(32000);
   if (pstack2==NULL)
        { printf("\nGrow region: Not enough memory\n");  return(-1); }

   px_push2(xc,yc);
   *(pimage2+(xc*im_columns)+yc)=3000;

   while(stack2)

        { px_pop2(&x,&y); i=(x-1)*im_columns; j=y;

          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) { *(pimage2+i+j)=3000;
                                          px_push2(x-1,y);     }
                else *(pimage2+i+j)=t;
/* printf("\n %d  (%d,%d)  %d",stack2, x-1,j, (int)*(pimage2+i+j)); */
          i=(x+1)*im_columns; j=y;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) {*(pimage2+i+j)=3000;
                                          px_push2(x+1,y);     }
                else *(pimage2+i+j)=t;

          i=x*im_columns; j=y-1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) { *(pimage2+i+j)=3000;
                                          px_push2(x,y-1);     }
                else *(pimage2+i+j)=t;

          i=x*im_columns; j=y+1;
          if (*(pimage2+i+j)==3000) i=i;
            else if (*(pimage2+i+j) <t) { *(pimage2+i+j)=3000;
                                          px_push2(x,y+1);     }
                else *(pimage2+i+j)=t;
        }

   x=0;
   for (i=0;i<im_rows*im_columns;i++)
        {       if (*(pimage2+i)!=3000 && *(pimage2+i)!=t) *(pimage2+i)=0;
                else  x++;     }
  free(pstack2);
  return(x);

}
int px_push2(x,y)
int x;
int y;
{
  *pstack2= (short int)x;
  pstack2++;
  *pstack2= (short int)y;
  pstack2++;
```

```
    stack2++;
   /* printf("\n %d  (%d,%d)",stack2, x,y);*/
   }

  int px_pop2(px, py)
   int *px;
   int *py;
   {
     short int i;
     pstack2--;
     i=*pstack2;
     *py=(int)i; pstack2--;
     i=*pstack2;
     *px=(int)i;
     stack2--;
   }


   /****************** Function to apply an adaptive threshold ******************/
   int adaptive_t(in_values,values,m_size,mask)
   short values[256][256],in_values[256][256],m_size,mask;
   {
   int i,j,k,x,y,sf=9;
   int sum1,sum2,sum3,max_val,t_bone=0;
   short values_x[256][256];

   max_val=maximum(in_values,m_size);
   printf("maximum value= %d\n",max_val);

   for (k=1; k<=mask; k++) {
           x=2;
           do {
           y=2;
           do {
           sum1=(values[x-1][y-1])+(values[x-1][y])+(values[x-1][y+1]);
           sum2=(values[x][y-1])+(values[x][y])+(values[x][y+1]);
           sum3=(values[x+1][y-1])+(values[x+1][y])+(values[x+1][y+1]);
           values_x[x][y]=(sum1+sum2+sum3)/sf;
          if (values_x[x][y] > 32000 )
                   values_x[x][y]=0;
           if (values_x[x][y] < 0) {
                   values_x[x][y]=0;
           }
           y=y+1;
           }
           while (y <=(m_size-1));
           x=x+1;
           }
           while (x<=(m_size-1));

           for (i=0; i<m_size; i++)
                   for (j=0; j<m_size; j++){
                           values[i][j]=values_x[i][j];
                   }
   }
   max_val=max_val-0.55*max_val;
           /*if (0.45*max_val) > 250) {
                   thresh_bone=max_;
           }
           else {
                   max_val=0.75*max_val;
           }*/
   /* apply an adaptive threshold to create a binary image */

           for (i=0;i<m_size;i++)
                   for(j=0;j<m_size;j++) {
```

```
  if (( in_values[i][j] > values_x[i][j])&&(in_values[i][j]>max_val)){
                                values[i][j]=255;
                                t_bone++;}
                        else
                                values[i][j]=0;
                }
return(t_bone);
}

/************** Function to check for node points  *************/
int nodes(in_values,values,m_size,mom)
short values[256][256],in_values[256][256],m_size;
int mom[5][5];
{
int x,y,node_sum,node=1,free_ends=0,x_m,y_m,x_dist,y_dist;
int i,j,isolated_points=0,network_area=0,m[9],e_dist;
double vector=0,vectors[5000],x_vect[5000],y_vect[5000];
double vector_sum,vector_sum1,vector_sum2,mean_vector,std_vector;
double std_x,std_y,sum_x=0,sum_y=0,x_mean,y_mean;

x_m=mom[1][0]/mom[0][0]; y_m=mom[0][1]/mom[0][0];
for(i=0; i<=5000; i++)
        vectors[i]=0;

for (i=0; i<m_size;i++)
for (j=0; j<m_size;j++)
        {
        if (in_values[i][j]==1)
                {network_area++;}
        }

        x=2;
        do
        {
        y=2;
        do
        {
         m[0]=in_values [x-1][y-1];
         m[1]=in_values[x-1][y];
         m[2]=in_values[x-1][y+1];
         m[3]=in_values[x][y-1];
         m[4]=in_values[x][y];
         m[5]=in_values[x][y+1];
         m[6]=in_values[x+1][y-1];
         m[7]=in_values[x+1][y];
         m[8]=in_values[x+1][y+1];
         node_sum=m[0]+m[1]+m[2]+m[3]+m[5]+m[6]+m[7]+m[8];
        if((m[4]==1)&&(node_sum>=3))
                {
                 values[x][y]=1000;
/* calculate vector */
                 e_dist=((x-x_m)*(x-x_m))+((y-y_m)*(y-y_m));
        x_vect[node]=(x-x_m); y_vect[node]=(y-y_m);
        sum_x=sum_x+x_vect[node]; sum_y=sum_y+y_vect[node];
                 vector=e_dist;
                 vectors[node]=sqrt(vector);
                node++;
                }
        if((m[4]==1)&&(node_sum==1))
                {
                 free_ends++;}
        if((m[4]==1)&&(node_sum==0))
                {
                 isolated_points++;}
        y=y+1;
        }
```

```
        while (y <=(m_size-1));
        x=x+1;
        }
        while (x<=(m_size-1));

vector_sum=0;
for (i=1; i<node; i++)
  vector_sum=vector_sum+vectors[i];

mean_vector=vector_sum/node;
x_mean=sum_x/node; y_mean=sum_y/node;

for (i=1; i<node; i++)
{
  vector_sum=vector_sum+((vectors[i]-mean_vector)*(vectors[i]-mean_vector));
 vector_sum1=vector_sum1+((x_vect[i]-x_mean)*(x_vect[i]-x_mean));
 vector_sum2=vector_sum2+((y_vect[i]-y_mean)*(y_vect[i]-y_mean));
}
std_vector=sqrt(vector_sum/node);
std_x=sqrt(vector_sum1/node);
std_y=sqrt(vector_sum2/node);

printf("means= %f  %f\n",x_mean,y_mean);
printf(" STD_X= %f\n",std_x);
printf(" STD_Y= %f\n",std_y);
printf(" Mean vector= %f\n",mean_vector);
printf(" STD vector= %f\n",std_vector);
printf("Network area= %d\n",network_area);
printf("Nodes= %d\n",node);
printf("Free ends= %d\n",free_ends);
printf("Isolated points=%d\n",isolated_points);
  return(node);
}


/**** Function  to implement the Euclidean distance transform ****/

int distance_transform(values,m_size)
short int values[256][256],m_size;
{
        int e1,e2,e3,e4,e5,min_pixel=0;
        int i,j;

/*** Apply Euclidean (2-3)distance transform  ***/
/** Forward raster scan **/

        i=2;
        do {
          j=2;
          do  {
               e1=2+values[i][j-1];
               min_pixel=e1;
               e2=3+values[i-1][j-1];
               if (e2 < min_pixel) {
                       min_pixel=e2;
               }
               e3=2+values[i-1][j];
               if (e3 < min_pixel) {
                       min_pixel=e3;
               }
               e4=3+values[i-1][j+1];
               if (e4 < min_pixel) {
                       min_pixel=e4;
               }
               e5=values[i][j];
               if (e5 == 0) {
```

```
                        min_pixel=0;
                }
                values[i][j]=min_pixel;
                j=j+1;
          }
          while (j <= m_size-1);
        i=i+1;
        }
        while (i <= m_size-1);


 /*** Reverse raster scan ***/

        i=m_size-1;
        do {
        j=m_size-1;
          do  {
                e1=2+values[i][j+1];
                min_pixel=e1;
                e2=3+values[i+1][j+1];
                if (e2 < min_pixel) {
                        min_pixel=e2;
                }
                e3=2+values[i+1][j];
                if (e3 < min_pixel) {
                        min_pixel=e3;
                }
                e4=3+values[i+1][j-1];
                if (e4 < min_pixel) {
                        min_pixel=e4;
                }
                e5=values[i][j];
                if (e5  < min_pixel) {
                        min_pixel=e5;
                }
                values[i][j]=min_pixel;
                j=j-1;
          }
          while (j >= 2);
        i=i-1;
        }
        while (i >= 2);

}

/* Routine to determine the number of regions connected by (i,j) */

int crossing_index(values,ii,jj)
short int values[256][256];
int ii,jj;
{
        int i,j,count;
        short int k;

        count=0;
/* start at position 8 */
        i=ii-1; j=jj-1; k=values[i][j];

/* move clockwise around (ii,jj), counting level changes */

        j++; /* move to (i-1,j) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        j++; /* move to (i-1,j+1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        i++; /* move to (i,j+1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
```

```
        i++; /* move to (i+1,j+1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        j--; /* move to (i+1,j) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        j--; /* move to (i+1,j-1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        i--; /* move to (i,j-1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        i--; /* move to (i-1,j-1) */
        if (k !=values[i][j] ){k=values[i][j]; count++;}
        return count/2;
}


/* Routine to determine the number of 8-nearest neigbbours to (i,j) */

int nay8(values,i,j,val)
short int values[256][256];
int i,j,val;
{
        int k;

        k=0;
        if(i < 1 || i >=255) return 0;
        if(j < 1 || j >=255) return 0;

        if(values[i][j]!= val) return 0;
        if(values[i-1][j]==val) k++;
        if(values[i-1][j+1]==val) k++;
        if(values[i][j+1]==val) k++;
        if(values[i+1][j+1]==val) k++;
        if(values[i+1][j]==val) k++;
        if(values[i+1][j-1]==val) k++;
        if(values[i][j-1]==val) k++;
        if(values[i-1][j-1]==val) k++;

        return k;
}


/* Zhang-Suen type of thinning procedure. Thin region labelled VAL */

int thinzs(values, val)
short int values[256][256];
int val;
{
        int i,j,n,again,bg;
        short int values2[256][256];

bg=0;

/* clear second image space to 0 */
        for(i=0; i<256; i++)
        {
                for(j=0; j<256; j++)
                        {values2[i][j]=0;}
        }

/* copy original image to second image space */

        for(i=0; i<256; i++)
        {
                for(j=0; j<256; j++)
                        {values2[i][j]=values[i][j];}
        }
```

```
do{

/* first pass through image */
        again=0;
        for(i=0; i<256; i++){
                for(j=0; j<256; j++)  {
                        if(values2[i][j]!=val) continue;
                        n=nay8(values2,i,j,val);
                        if( (n >=2 ) && (n <=6))  {
                          if(crossing_index(values2,i,j)==1)  {
                            if((values2[i-1][j]==bg)||
                                        (values2[i][j+1]==bg)||
                                        (values2[i+1][j]==bg)) {
                                if((values2[i][j+1]==bg)||
                                        (values2[i+1][j]==bg)||
                                        (values2[i][j-1]==bg)) {
                                values[i][j]=bg;
                                again=1;
                                }
                             }
                           }
                        }
                        }
        }

        /* copy original image to second image space */

        for(i=0; i<256; i++)
                for(j=0; j<256; j++)
                        {values2[i][j]=values[i][j];}


/* second pass through image */

        for(i=0; i<256; i++){
                for(j=0; j<256; j++)  {
                        if(values[i][j]!=val) continue;
                        n=nay8(values,i,j,val);
                        if( (n >=2 ) && (n <=6))  {
                          if(crossing_index(values,i,j)==1)  {
                            if((values[i-1][j]==bg)||
                                        (values[i][j+1]==bg)||
                                        (values[i][j-1]==bg)) {
                                if((values[i-1][j]==bg)||
                                        (values[i+1][j]==bg)||
                                        (values[i][j-1]==bg)) {
                                values2[i][j]=bg;
                                again=1;
                                }
                             }
                           }
                        }
                        }
        }


        /* copy second image to original image space */

        for(i=0; i<256; i++)
        {
                for(j=0; j<256; j++)
                        {values[i][j]=values2[i][j];}
        }
} while (again);
```

```
}


/** Function to find local maximum and saddle points from distance transform **/

int local_maximum(values,out_values)
        short int values[256][256],out_values[256][256];
{
        short int local_max[256][256],saddle[256][256];
        int e1,e2,e3,e4,e5,e6,e7,e8,e9,min_pixel,max_pixel;
        int x,y,l_max,sadl;



/*  initalize values matrix to 0 */

     for (x=1; x<255; x++)
        for (y=1; y<255; y++) {
                local_max[x][y]=0;
                saddle[x][y]=0;
                out_values[x][y]=0;
        }


/*** Find local maxima ***/
l_max=0;

    for (x=1; x<255; x++)
        for (y=1; y<255; y++) {
                if (values[x][y] == 0) continue;
                e1=values[x][y-1];
                max_pixel=e1;
                e2=values[x-1][y-1];
                if (e2 > max_pixel){
                        max_pixel=e2;
                }
                e3=values[x-1][y];
                if (e3 > max_pixel){
                        max_pixel=e3;
                }
                e4=values[x-1][y+1];
                if (e4 > max_pixel){
                        max_pixel=e4;
                }
                e5=values[x][y+1];
                if (e5 > max_pixel){
                        max_pixel=e5;
                }
                e6=values[x+1][y+1];
                if (e6 > max_pixel){
                        max_pixel=e6;
                }
                e7=values[x+1][y];
                if (e7 > max_pixel){
                        max_pixel=e7;
                }
                e8=values[x+1][y-1];
                if (e8 > max_pixel){
                        max_pixel=e8;
                }
                e9=values[x][y];
                if ( e9 >=  max_pixel){
                        local_max[x][y]=100;
                        l_max++;
                }
      }
```

```
 printf("local max found= %d\n",l_max);

 /** check for saddle points **/
 sadl=0;
 for (x=1; x<255; x++)
        for (y=1; y<255; y++) {
                e9=values[x][y];
                if (e9==0) continue;
                if (local_max[x][y] ==100) continue;
                e1=values[x][y-1];
                e2=values[x-1][y-1];
                e3=values[x-1][y];
                e4=values[x-1][y+1];
                e5=values[x][y+1];
                e6=values[x+1][y+1];
                e7=values[x+1][y];
                e8=values[x+1][y-1];
                if ((e9==e1)||(e9==e2)||(e9==e3)||(e9==e4)) continue;
                        if((e1==e5)||(e2==e6)||(e3==e7)||(e4==e8)){
                                local_max[x][y]=100;
                                saddle[x][y]=200;
                                sadl++;
                        }
        }

printf("saddles found= %d\n",sadl);


for (x=1; x<255; x++)
        for (y=1; y<255; y++) {
                out_values[x][y]=local_max[x][y];
        }



}

/************** Function to check for node points  **************/
int indices(in_values,values,rows,columns,val,nd_label,fe_label,res_file)
short values[256][256],in_values[256][256];
int rows,columns,nd_label,fe_label,val;
FILE *res_file;
{
int x,y,node_sum,node=0,free_ends=0;
int i,j,isolated_points=0,network_length=0,m[9];
int k,count;
double CI;
for(i=0; i<=9; i++)
        m[i]=0;      /* clear m to 0 */

for (i=0; i<rows;i++)
        for (j=0; j<columns;j++){
                values[i][j]=0;    /* clear values to 0 */
                if (in_values[i][j]==val){
                        values[i][j]=100;
                        network_length++;
                }
        }

for (x=2; x<=(rows-1); x++)
        for (y=2; y <= (columns-1); y++) {
/* first make sure that two adjacent points are not both identifies as nodes */
                if (values[x][y]==0) continue;
                if (values [x-1][y-1]==nd_label) continue;
                if (values[x-1][y]==nd_label) continue;
                if(values[x-1][y+1]==nd_label) continue;
```

```
                    if (values[x][y-1]==nd_label) continue;
                    if(values[x][y+1]==nd_label) continue;
                    if (values[x+1][y-1]==nd_label) continue;
                    if (values[x+1][y]==nd_label) continue;
                    if (values[x+1][y+1]==nd_label) continue;

    /* start at position 8 */
                    count=0; i=x-1; j=y-1; k=values[i][j];
    /* move clockwise around (ii,jj), counting level changes */

                    j++; /* move to (i-1,j) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    j++; /* move to (i-1,j+1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    i++; /* move to (i,j+1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    i++; /* move to (i+1,j+1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    j--; /* move to (i+1,j) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    j--; /* move to (i+1,j-1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    i--; /* move to (i,j-1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}
                    i--; /* move to (i-1,j-1) */
                    if (k !=values[i][j] ){k=values[i][j]; count++;}

                    if(count >=6) {
                            values[x][y]=nd_label; /* label nodes  */
                            node++;
                    }
            }

    /* Score for free ends and isolated points */
    node_sum=0;
    for (x=2; x<=(rows-1); x++)
            for (y=2; y <= (columns-1); y++) {
                    m[0]=in_values [x-1][y-1];
                    m[1]=in_values[x-1][y];
                    m[2]=in_values[x-1][y+1];
                    m[3]=in_values[x][y-1];
                    m[4]=in_values[x][y];
                    m[5]=in_values[x][y+1];
                    m[6]=in_values[x+1][y-1];
                    m[7]=in_values[x+1][y];
                    m[8]=in_values[x+1][y+1];
                    node_sum=m[0]+m[1]+m[2]+m[3]+m[5]+m[6]+m[7]+m[8];
                            if((m[4]>0)&&(node_sum==val)){
                                    values[x][y]=fe_label; /* label free ends  */
                                    free_ends++;
                            }
                            if((m[4]>0)&&(node_sum==0)) {
                                    isolated_points++;
                            }
                            if((m[4]>0)&&(m[5]==val)&&(m[7]==val)&&(m[8]==val)) {
                                    values[x][y]=nd_label; /* label nodes  */
                                    node++;
                            }

            }
CI=(((double) node-(double) free_ends-(double) isolated_points)/((double) network_length))*
printf("Network length= %d\n",network_length);
printf("Nodes= %d\n",node);
printf("Free ends= %d\n",free_ends);
printf("Isolated points=%d\n",isolated_points);
printf("Connectivity Index=%.2f\n",CI);
```

```
fprintf(res_file,"Network length= %d\n",network_length);
fprintf(res_file,"Nodes= %d\n",node);
fprintf(res_file,"Free ends= %d\n",free_ends);
fprintf(res_file,"Isolated points=%d\n",isolated_points);
fprintf(res_file,"Connectivity Index=%.2f\n",CI);
/*return(node);*/
}



/*** Routine to count holes in binary image  ***/

int hole_counter(h,values,rx1,rx2,ry1,ry2)
short int values[256][256],rx1,rx2,ry1,ry2;
int h[1000];
{
int i,j,count=0,val=1;


for(i=0; i<1000; i++)
        h[i]=0;      /* clear h to 0 */

for (i=ry1+2; i<ry2;i++)
        for (j=rx1+2 ; j<rx2;j++){
                if (values[i][j] !=0)  continue;
                        h[count]=fill_hole (values,256,256,i,j,0,val);
                        printf("# of holes= %d   %d\n",count,h[count]);
                        count++;
                        val +=1;
        }
return count;
}


/******* Function fill hole by region grow given a seed pixel **********/

int stack3=0;
short int *pstack3;

int fill_hole (pimage,im_rows,im_columns,xc,yc,t,val)
short *pimage;
int im_rows,im_columns;
int xc,yc;
short t,val;
{
   int i,j,x,y;
    short *pbuf;

   pstack3 = (short int *) malloc(32000);
   if (pstack3==NULL)
        { printf("\nGrow region: Not enough memory\n");  return(-1); }

   px_push3(xc,yc);
   *(pimage+(xc*im_columns)+yc)=val;

   while(stack3)
        {
          px_pop3(&x,&y);

         i=(x-1)*im_columns; j=y;
          if (*(pimage+i+j)==val) i=i;
            else if (*(pimage+i+j)<=t) { *(pimage+i+j)=val;
                                         px_push3(x-1,y);     }
                else *(pimage+i+j)=*(pimage+i+j);

         i=(x+1)*im_columns; j=y;
```

```
            if (*(pimage+i+j)==val) i=i;
              else if (*(pimage+i+j)<=t) {*(pimage+i+j)=val;
                                          px_push3(x+1,y);     }
                   else *(pimage+i+j)=*(pimage+i+j);

            i=x*im_columns; j=y-1;
            if (*(pimage+i+j)==val) i=i;
              else if (*(pimage+i+j)<=t) { *(pimage+i+j)=val;
                                          px_push3(x,y-1);     }
                   else *(pimage+i+j)=*(pimage+i+j);

            i=x*im_columns; j=y+1;
            if (*(pimage+i+j)==val) i=i;
              else if (*(pimage+i+j)<=t) { *(pimage+i+j)=val;
                                          px_push3(x,y+1);     }
                   else *(pimage+i+j)=*(pimage+i+j);
          }


    x=0;
    for (i=0;i<im_rows*im_columns;i++)
          {      if (*(pimage+i)!=val) *(pimage+i)=*(pimage+i);
                 else  x++;     }
   free(pstack3);
   return(x);
 }

 int px_push3(x,y)
 int x;
 int y;
 {
   *pstack3= (short int)x;
   pstack3++;
   *pstack3= (short int)y;
   pstack3++;
   stack3++;
  /*printf("\n %d  (%d,%d)",stack3, x,y);*/
 }

 int px_pop3(px, py)
 int *px;
 int *py;
 {
   short int i;
   pstack3--;
   i=*pstack3;
   *py=(int)i; pstack3--;
   i=*pstack3;
   *px=(int)i;
   stack3--;
 }


/****** Function to calculate mean and standard deviation of an array *****/
int mean_std(values,m,std,count,pixel_size,res_file)
int values[1000],count;
float m, std,pixel_size;
FILE *res_file;
{
int i,sum=0;
double sum_std=0.0;
for (i=1; i<= count; i++)  /* calculate mean */
        {sum +=values[i];}
m=(sum/((float)count))*(pixel_size)*(pixel_size);
printf("\n MEAN =%.2f mm2\n", m);
```

```
  fprintf(res_file,"Number of holes detected= %d\n",count);
  fprintf(res_file,"Mean hole area=%.2f mm2\n", m);

  for (i=1;i<=count;i++) {        /* calculate mean for holes with area >1 pixel */
          sum_std=sum_std+((values[i]-m)*(values[i]-m));}
          sum_std=sum_std/(count-1);
          std=sqrt(sum_std)*(pixel_size)*(pixel_size);
          /*printf("STD =%.2f mm2\n", std);*/
  }


  /****** Function to sort an array of integers in ascending order *******/
  int sort(a,n,pixel_size,res_file)
  int a[1000],n;
  float pixel_size;
  FILE *res_file;
  {
          int i,j,temp,sum=0,sum1=0;
          float min,max,med,max1,max2,max3,m;
          for (i=0; i<=(n-1); ++i)
                  for (j=i+1; j<=n; ++j) {
                          if (a[i]>a[j]) {
                                  temp=a[i];
                                   a[i]=a[j];
                                   a[j]=temp;
                          }
                  }
          for (i=0; i<=n ; i++)  /* calculate mean */
          {sum +=a[i];}
          m=(sum/((float)n))*(pixel_size)*(pixel_size);
          /* printf("\n MEAN1 =%.2f mm2\n", m);*/

          for (i=0; i<=n-1 ; i++)  /* calculate mean */
          {sum1 +=a[i];}
          m=(sum1/((float)n-1))*(pixel_size)*(pixel_size);
          /* printf("\n MEAN2 =%.2f mm2\n", m);*/

          min=(float)(a[0])*(pixel_size)*(pixel_size);
          max=(float)(a[n-1])*(pixel_size)*(pixel_size);
          med=(float)(a[n/2])*(pixel_size)*(pixel_size);
          max1=(float)(a[n-2])*(pixel_size)*(pixel_size);
          max2=(float)(a[n-3])*(pixel_size)*(pixel_size);
          max3=(float)(a[n-4])*(pixel_size)*(pixel_size);

          printf("Median= %.2f mm2\n", med);
          printf("Minimum= %.2f mm2\n", min);
          printf("Maximum = %.2f mm2\n", max);
          printf("Maximum-1= %.2f mm2\n", max1);
          printf("Maximum-2= %.2f mm2\n", max2);
          printf("Maximum-3= %.2f mm2\n", max3);


          fprintf(res_file,"Median= %.2f mm2\n", med);
          fprintf(res_file,"Minimum= %.2f mm2\n", min);
          fprintf(res_file,"Maximum = %.2f mm2\n", max);
          fprintf(res_file,"Maximum-1= %.2f mm2\n", max1);
          fprintf(res_file,"Maximum-2= %.2f mm2\n", max2);
          fprintf(res_file,"Maximum-3= %.2f mm2\n", max3);

  }

  /****** Function to find hole size distribution *******/
  int hole_distribution(a,n,bin)
  int a[1000],bin,n;
  {
          int i,j,temp;
```

```
        int h[750];
        char *buf2,*fpoint2;
        char out_file[25];
        FILE *out;

 /* clear histogram */
        for (i=0; i<=(n-1); ++i){
                h[i]=0;
        }
        for (i=0; i<=(n-1); ++i) {
                h[a[i]/bin]=h[a[i]/bin]+1;
        }

 /* *** write to file in ascii format *** */
        buf2=(char*) malloc(5000);
        printf("Enter output file:> ");
        scanf("%24s",out_file);
        if ((out=fopen(out_file,"w+t")) == (FILE *) NULL){
                printf("File Error");
                exit(1);
        }
        fpoint2=(char*) buf2;
        for(i=0; i<=200; ++i){
                sprintf(fpoint2, "%d  %d\n",i,h[i]);
                fputs (fpoint2, out);
        }
free(buf2);
}

int dilation(pimage, n_bytes, im_rows,im_columns, pelem,elem_rows,elem_columns)
int n_bytes, im_rows, im_columns, elem_rows, elem_columns;
unsigned char *pimage, *pelem;
{  int i,j,x,y, h_rows, h_col, sum, itemp, total, el_temp, im_temp, value;
   short int  ival, *pbi, *pimage2, *pi;
   unsigned char cval, *pbuf, *pcimage2, *pci, ctemp;

   if (n_bytes==1) pbuf = pimage;
        else if (n_bytes==2) pbi = (short int *) pimage;
          else return(-2);

   total=im_rows*im_columns;  ival=0; cval=0; j=0;
   do { if (n_bytes==2) ival=*pbi++;
              else cval=*pbuf++;  j++; } while(!ival && !cval && j<total);
/* printf("\n first no-zero value at %d  (%d  %d)", j, im_rows, im_columns); */
   if (!ival && !cval) return(-3);

   total=im_rows*im_columns*n_bytes;
   pcimage2=(unsigned char *) malloc(total);     /* get memory for new image  */
   if (pcimage2==NULL) return(-1);
   pimage2=(short int *) pcimage2;

   pbuf=pcimage2; for(i=0;i<total;i++)  *pbuf++=0;

   h_rows=elem_rows/2; h_col=elem_columns/2;

   pbi = (short int *) pimage;
   x=h_rows;
   do { y=h_col;
        do { sum=0;
             pbuf=pelem;
             for (i=0;i<elem_rows;i++)
                for (j=0;j<elem_columns;j++)
                   { el_temp=(int) (*pbuf); pbuf++;
                     if (n_bytes==2)
                            im_temp=(int) (*(pbi+(x-h_rows+i)*im_columns
                                                +y-h_col+j) );
```

```
                    else  im_temp=(int) (*(pimage+(x-h_rows+i)*im_columns
                                                +y-h_col+j) );
                    itemp = el_temp*im_temp;
                    sum += itemp;  }

            if (sum) {  if (n_bytes==2)
                              *(pimage2+x*im_columns+y) = ival;
                         else
                              *(pcimage2+x*im_columns+y) = cval;
                       }
            y++;
           } while (y < im_columns-h_col);
         x++;
       }  while (x < im_rows-h_rows);

   pbuf=pimage;  pci=pcimage2; j=0;
   pbi=(short int *) pimage; pi=pimage2;
   total=im_rows*im_columns*n_bytes;
   for(i=0;i<total;i++) *pbuf++ = *pci++;

/*    { if (n_bytes==2)
         { if (*pi && !j)
           { printf(" \n first no zero at  %d (%d)  ", (int)*pci, i);  j=1; }
          *pbi++=*pi++;  }
       else
         { if (*pci && !j)
           { printf(" \n first no zero at  %d (%d)  ", (int)*pci, i);  j=1; }
          *pbuf++ = *pci++;  }
    }
*/
   value=ival + (int)cval;
   free(pcimage2);
   return(value);
}



int erosion(pimage, n_bytes, im_rows,im_columns, pelem,elem_rows,elem_columns)
int n_bytes, im_rows, im_columns, elem_rows, elem_columns;
unsigned char  *pimage, *pelem;
{  int i,j,x,y, h_rows, h_col, sum, itemp, total, el_temp, im_temp, value;
   int el_sum;
   short int  ival, *pbi, *pimage2;
   unsigned char cval, *pbuf, *pcimage2, *pci, ctemp;

   if (n_bytes==1) pbuf = pimage;
        else if (n_bytes==2) pbi = (short int *) pimage;
          else return(-2);

   i=im_rows*im_columns;  ival=0; cval=0;
   do { if (n_bytes==2) ival=*pbi++;
                else cval=*pbuf++; i--; } while(!ival && !cval && i);
   if (!ival && !cval) return(-3);
   value=ival + (int)cval;

   total=im_rows*im_columns*n_bytes;
   pcimage2=(unsigned char *) malloc(total);     /* get memory for new image  */
   if (pcimage2==NULL) return(-1);
   pimage2=(short int *) pcimage2;

   pbuf=pcimage2; for(i=0;i<total;i++)  *pbuf++=0;
   pbuf=pelem;  el_sum=0;
   for(i=0;i<elem_rows*elem_columns;i++)
        { j=(int) (*pbuf);  el_sum += j; pbuf++;  }
   el_sum = el_sum * value;
```

```
  /*  printf("\n ival=%d   cval=%d  value=%d  el_sum=%d\n", ival, (int)cval,
                  value, el_sum);
  */
     h_rows=elem_rows/2; h_col=elem_columns/2;
     pbi = (short int *) pimage;
     x=h_rows;
     do { y=h_col;
          do { sum=0;
               pbuf=pelem;
               for (i=0;i<elem_rows;i++)
                  for (j=0;j<elem_columns;j++)
                     { el_temp=(int) (*pbuf); pbuf++;
                       if (n_bytes==2)
                               im_temp=(int) (*(pbi+(x-h_rows+i)*im_columns
                                                  +y-h_col+j) );
                          else  im_temp=(int) (*(pimage+(x-h_rows+i)*im_columns
                                                  +y-h_col+j) );
                       itemp = el_temp*im_temp;
                       sum += itemp;  }
  /* if (sum) printf("\n %d   %d   %d   ", x,y,sum);  */
               if (sum == el_sum)
                     {    if (n_bytes==2)
                                *(pimage2+x*im_columns+y)=ival;
                          else
                                *(pcimage2+x*im_columns+y)=cval;
                       }
               y++;
             } while (y < im_columns-h_col);
          x++;
        }  while (x < im_rows-h_rows);

     pbuf=pimage;  pci=pcimage2;
     for(i=0;i<total;i++)  *pbuf++ = *pci++;
  /*      {  if (*pci) printf("  %d (%d)  ", (int)*pci, i);
            *pbuf++ = *pci++;  }
  */
     free(pcimage2);
     return(value);

  }


  int morph_open(pimage, n_bytes, im_rows,im_col, pelem,elem_rows,elem_col)
  int n_bytes, im_rows, im_col, elem_rows, elem_col;
   unsigned char *pimage, *pelem;
  {  int i;

          i=erosion(pimage, n_bytes, im_rows,
                          im_col, pelem,elem_rows,elem_col);
      if (i>0)
          i=dilation(pimage, n_bytes, im_rows,
                          im_col, pelem,elem_rows,elem_col);
      return(i);
  }



  int morph_close(pimage, n_bytes, im_rows,im_col, pelem,elem_rows,elem_col)
  int n_bytes, im_rows, im_col, elem_rows, elem_col;
  unsigned char *pimage, *pelem;
  {  int i;

      i=dilation(pimage, n_bytes, im_rows,im_col,
                          pelem,elem_rows,elem_col);
      if (i>0)
          i=erosion(pimage, n_bytes, im_rows,im_col,
```

```
                    pelem,elem_rows,elem_col);
    return(i);
}
;
```