Semi-Automated Analysis of the Texture of Osteoarthritis in Trabecular Bone

by Jason Nochlin

An Honors Thesis for the
Department of Electrical Engineering

Jason Nochlin

Dr. Eric Miller
Department of Electrical Engineering
Tufts University

Dr. Timothy McAlindon, MD
Division of Rheumatology
Tufts Medical Center

Dr. Ronald Lasser
Department of Electrical Engineering
Tufts University

Contents

ABSTRACT

In digital images, texture refers to patterns in the intensity of pixels of the image. By analyzing the texture of an image, it is possible to generate data about the subpixel structure of the image's subject. Specifically, in the problem of diagnosing the severity of osteoarthritis (OA) in the periartricular tibia (i.e., bone just under the knee joint), texture analysis can be used to quantify changes in the trabecular structure of the bone even when the trabeculae are not resolved in the MRI image. Texture-based techniques provide medical researchers a tool for understanding OA and how it changes over time.

To explore the utility of texture techniques, a process was developed which predicts OA severity relative to an existing benchmark, the relative perarticular Bone Mineral Density (paBMD) as determined by dual-energy X-ray absorptiometry (DXA). This processed was developed and tested using a dataset consisting of knee MRI images from 50 patients.

After exploring a variety of methods, a three step process was implemented. First, image segmentation is performed in order to identify the region of interest (ROI) in the image with minimal human input, a task previously performed entirely manually. Second, spatial filters are applied to each ROI in order to generate texture maps. A bank of 40 filters was used which consisted of two classes of filters: Laws texture energy masks and Gabor filters. Statistics are taken on the texture maps (mean, standard deviation, skewness, kurtosis, entropy). Third, the statistics are then used as features for a regression problem. To solve the regression problem, kernel-Support Vector Regression with a 4th degree polynomial was found to be the best performing algorithm.

In the final analysis, the knee images of 39 patients were used in a cross-validation scheme. A prediction root-mean square error of 11% was obtained, suggesting that texture analysis techniques can be used as tools for analyzing trabecular structure in MRIs.

Chapter 1

Introduction

This report describes the research project aimed at developing a set of algorithms to aid the semi-automatic processing of Magnetic Resonance Images (MRI) of knees in a medical research study being pursued at the Tufts Medical Center in the Department of Rheumatology. The study aims to develop methods for quantifying the severity of osteoarthritis using medical imaging techniques. This research was conducted as part of a collaborative effort with the Electrical Engineering Department at Tufts University.

The medical study consisted of monitoring the progression of osteoarthritis in the knees of several hundred patients. The doctors involved in the study are exploring how the periarticular (near joint) bone participates in the progression of osteoarthritis. The knees of each patient in the study were imaged using Magnetic Resonance Imaging (MRI). The doctors then use the MRIs to obtain information about the mineral composition and trabecular structure of the bones around the knee.

The initial goal of the research was to explore the use of digital image processing methods to generate information from the MRI that the doctors currently must gather through other methods. Over the course of researching and testing processing techniques, several image processing concepts were used to locate specific regions in the MRI images and to generate data describing the texture of that region. Machine learning (i.e., computational statistics) tools were then used to analyze these data and benchmark the results against other data from the study.

The initial results from the process are promising and provide evidence that the methods used could provide valuable diagnostic information to doctors and medical researchers.

1.1 Medical Problem Overview

Osteoarthritis (OA), also known as degenerative joint disease, is a condition that occurs when the cartilage between bones breaks down, resulting in increased stress on the bones. This can lead to a variety of structural deformities of the bone. OA gradually worsens with time. Treatments exist, but they only relieve the symptoms and do not fix the condition. The doctors in the Division of Rheumatology at Tufts Medical Center are specifically interested in OA of the knee and are conducting leading research in this field [2, 5].

The collaborative efforts where engineering tools have been applied were focused on one particular region in the knee: the subchondral bone of the periarticular tibia. Subchondral refers to bone in the region directly under the rounded end of a long bone. Periarticular means the area of

the bone near a joint, in this case the knee. The tibia is one of two bones in the lower leg. In this region, there is a particular interest in studying how changes or deformations in the structures that make up bone (trabeculae) relate to OA.

In addition to MRI images, the department also has data concerning bone mineral density of the region of interest, in medical terms, the tibial paBMD. These data were obtained using another imaging technique known as dual x-ray absorptiometry (DXA). The ratio of the bone mineral density on one side of the knee to the other (medial:lateral paBMD ratio) is used as a baseline measure of the degree of OA for a patient, as numerous studies indicate its construct validity. This is not a perfect measure for the severity of OA, but it is used as a simple way to verify the effectiveness of new diagnostic techniques.

Each pixel in the MRIs had dimensions of .2 x .2mm. At this resolution, it is not possible to observe the detailed structure of the trabeculae. Because of this, texture algorithms were used as a alternative means of quantifying the structure. By analyzing the texture (a.k.a. repeating patterns) of an MRI image, patterns can be identified that have some correlation with the underlying structure.

1.2 Engineering Solution Overview

The raw MRI image data from the right knees of 50 patients were used as a sample for developing image processing tools for bone analysis. The final analysis was used on 39 knees, the software failed to function on the other 11 knees. Further refinements to the algorithm or the creation of a manual processing method would allow for the analysis of the failed knees. After analyzing the images, the resulting statistics were used to find methods that correlated well to the relative DXA values for each patient.

The image analysis consisted of two major steps. First, image segmentation was used to identify the region of interest in each MRI image. Second, texture filters were applied to each image. The texture filters are used to recognize repeating patterns in the image. These filters effectively transform an image into texture maps.

Statistics were then computed from the texture map, which created a vector of values that summarized the texture of a particular image as described by a particular texture filter. Numerous texture filters were used on each image.

After analyzing the images, the sample data were used in a regression problem to look for the statistics that showed the strongest correlation with relative DXA. The texture filters that were used to gather those statistics should be the ones that describe texture patterns that best discriminate between OA and non-OA knees.

1.3 Significance of Research

This research and the resulting software tool provide a novel means for gaining insights into a medical problem. They provide a means to measure how changes in the periarticular bone participate in the structural progression of knee OA, which aid in the development of new treatments and early detection schemes. This thesis can serve as a basis for further research and collaboration.

As noted in Section 1.2, the software implementation used is currently still in the prototype stage. Suggestions for implementation improvements and further research are included in Chapter 4.

1.4 Outline of Remainder of the Thesis

The remainder of the thesis discusses the details of the theoretical concepts required to build the bone analysis tools, how these concepts were implemented in software, the experimental results, and suggestions for future work.

Specifically, Chapter 2 covers the background material required for understanding and applying the concepts in digital image processing and machine learning that were used to build the tools. Chapter 3 contains the implementation details for the final process that analyzes bones, generates data, and performs the machine learning analysis. Chapter 4 contains the conclusion, results, and suggestions for further research. The appendix contains example data and the MATLAB implementation code for the tools.

Chapter 2

Background

2.1 Image Processing

An image is a two-dimensional spatial representation of some object or setting in nature. People have always had an interest in manipulating and analyzing images. In the past half-century, computers have given us the ability to perform extremely complex operations in images to achieve a variety of goals.

A standard film photograph provides an analog representation of an image. The image exists in a continuous space and colors are represented on a continuous spectrum. In order to work with an image on a computer system, a discrete representation must be obtained by sampling the original image. This necessity gave rise to the pixel, a discrete representation of color, which is the most basic component of a digital image. A pixel consists of either a single numerical value, as in the case of binary and grayscale images, or an array of values, as in the case of color images, with an associated location (see Figure 2.1). With color images, instead of having pixels with an array of values, the image is often thought of as having multiple color planes. Digital image processing is the field of study concerned with the manipulation and analysis of digital images [6].

Digital images are usually composed of a rectangular grid of pixels with a fixed width and height. This field uses a variety of techniques from signal processing, mathematics, and computer science. For easy manipulation, the rectangular array of pixels is often represented in the computer as an $m \times n$ matrix, where m is the height of the image (i.e., number of rows) and n is the width (i.e., number of columns). Pixel values are usually represented as a binary, integer, or truncated real value. Each pixel can have one value (such as in a grayscale image) or an ordered array of values (such as in a color image).

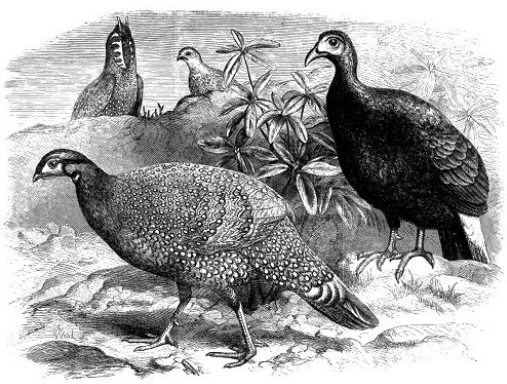
2.1.1 Motivations and Applications for Digital Image Processing

Converting an image into a digital form allows many types of operations and analysis to be performed on the image. Some specific algorithms that are used in this project are discussed in Section 2.1.2.

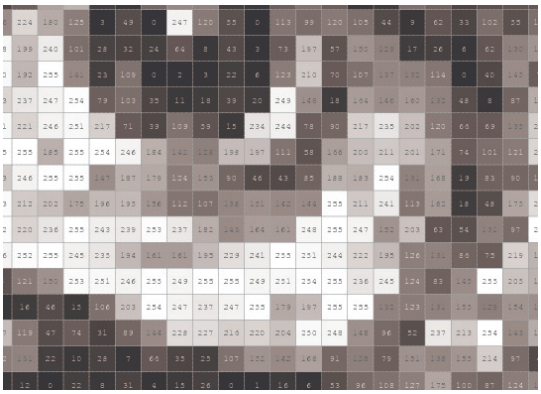
Image processing methods are designed for one or more of the following three general objectives: transformation, recognition, and analysis. Other operations exist outside of these categories, but these provide a fairly comprehensive overview of the field.

Transformations include a variety of operations that change the appearance of an image. Some simple examples are color-to-grayscale conversion, shrinking/enlarging, and cropping. More com-

Figure 2.1: Example of an image converted into computer form



(a) A grayscale image



(b) An image matrix taken from a small region of the example image

plex transformations would include de-noising (filtering out unwanted interference to clarify the image) and morphing (combining two images).

Recognition includes any process that attempts to locate or track specific regions in an image. For example, in this project, recognition methods were used to find the bone region of an MRI image. These methods are commonly used to segment an image in which specific objects or larger regions in an image are identified. Recognition methods also can be used to test for the presence of some feature in an image.

Analysis includes any process that attempts to extract data from an image. This a broad category and can involve methods from several fields. Analysis tasks range from discrete object selection (i.e., finding some form in an image) to statistical analysis of information in the image. Many of the techniques in this category are generalizations of techniques used to analyze one-dimensional signals (an image can be thought of as a two-dimensional signal). For example, frequency analysis (Fourier Transforms) can be applied to gather information about the spatial structure of an image.

2.1.2 Algorithms Introduction

All images included in this project were either grayscale or binary. Grayscale implies that the entries in the image matrix were positive integer values, corresponding to the intensity of the image at that point. In the case of the dataset for this project, the pixel values were integers between 1 and 1,024. Lower values were less intense (1 corresponded to black), while higher values were more intense (1,024 corresponded to white). In some cases, image operations resulted in non-integer values being used as pixel data. For example, when taking the average value of a neighborhood of integer value pixels, it is common for the result to be non-integral.

Binary images were used at intermediate stages in this project. A binary image is simply a matrix where the entries are logical values (i.e., true/false or 0/1). Binary images are commonly used to show individual pixels or connected regions in an image that have some property.

2.1.3 Spatial Filtering

One method for identifying repeating patterns in an image (texture analysis) is to apply a variety of discrete filters to the image that have varying frequency response properties. These can be low, high, or band pass filters, similar to their one-dimensional analogues, or the filters can have any other frequency response. To apply these filters, the convolution operation is used.

Convolution

Convolution is an operation that combines two input signals into one output signal. The one-dimensional and two-dimensional formulas are:

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]\, g[n-m] \tag{2.1}$$

$$(f * g)[n_1, n_2] = \sum_{m_1=-\infty}^{\infty} \sum_{m_2=-\infty}^{\infty} f[m_1, m_2]\, g[n_1 - m_1, n_2 - m_2] \tag{2.2}$$

Convolution essentially takes one signal and shifts it over the entire duration of another signal. With each shift, a new value is computed by multiplying every point on one signal with the value of the other signal that is currently at the same point and then sum over the entire domain. Convolution can be used to extract texture information from an image.

For analyzing texture, a small image (relative to the size of the whole image) with a specific texture content can be convolved with a grayscale image. These small images are referred to as *filters* or *masks*. The result of the convolution operation is a new image where the intensity values are positive and negative integers or real values in the case where the values of the texture mask are non-integral. The value of the image at any point reflects the "amount" of the texture (as specified by the structure of the mask) in the image at that point.

Texture Filters

Two classes of texture filters were used: Laws texture energy measures [10] and Gabor filters [8].

The Laws texture energy measures, an example of which is shown in Figure 2.2, are derived from a set of five, five-pixel long, one-dimensional filters that are combined using an outer product in all combinations to form 25 two-dimensional texture filters. Each of the five basic filters is designed to detect a repeating pixel pattern in a one-dimensional signal. When a combination is created, one filter is aligned vertically and another horizontally. The resulting two-dimensional filter detects the simultaneous presence of one pattern vertically in an image and another horizontally in the image.

Gabor filters are two-dimensional structures created by combining a cosine wave moving in one direction and a two-dimensional Gaussian distribution that acts as a window that selects only a certain portion of the cosine wave. An example of a Gabor filter is shown in Figure 2.3. Gabor filters have five parameters: wavelength of the cosine wave in pixels (λ), orientation of the cosine wave (θ), phase of the cosine wave (ϕ), standard deviation of the Gaussian in the horizontal direction (σ_x), and standard deviation of the Gaussian in the vertical direction (σ_y). The equation for a Gabor filter is [8]:

$$g(x, y; \lambda, \theta, \phi, \sigma_x, \sigma_y) = e^{\frac{-x\cos\theta - y\sin\theta}{\sigma_x^2} + \frac{x\sin\theta - y\cos\theta}{\sigma_y^2}} \cos(2\pi \frac{x\cos\theta + y\sin\theta}{\lambda} + \phi) \tag{2.3}$$

Figure 2.2: Example of a Laws texture mask.

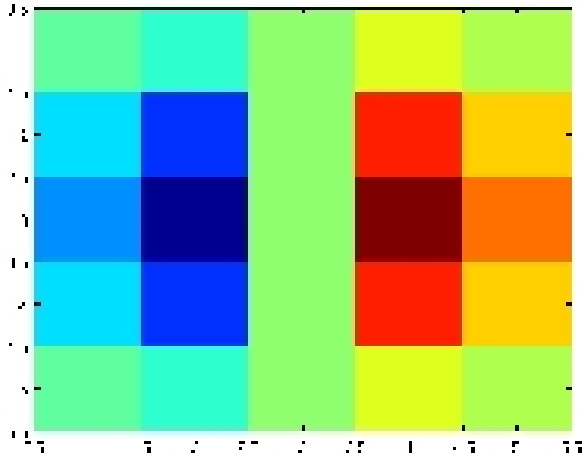
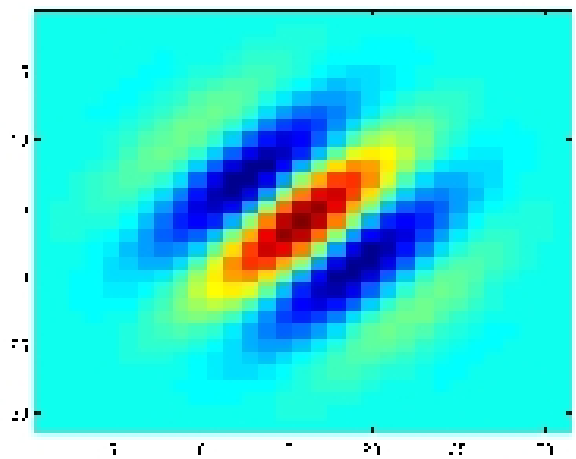


Figure 2.3: Example of a Gabor filter, scaled to show detail.



Thresholding

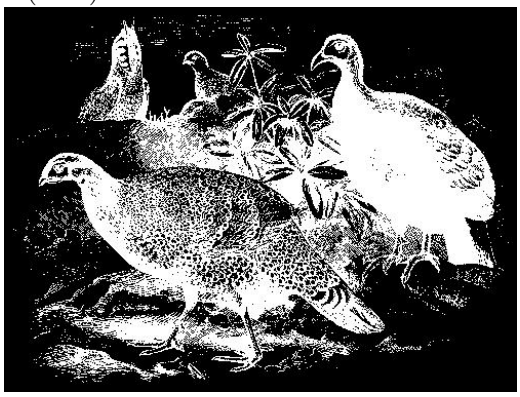
One of the most basic image processing methods is binary thresholding. For grayscale images, this technique applies a logical greater than (or less than) operation to the image and outputs a binary matrix based on the result of that logical operation for each pixel.

Thresholding essentially selects all pixels in an image with intensity greater than (or less than) some specified value [6]. Multiple thresholding operations on the original image can be used to select pixels in one or more desired ranges. A single thresholding operation can be defined by the formula:

$$I^*_{i,j} \quad = \quad I_{i,j} \,\square\, T \tag{2.4}$$

where I is the original image, I^* is the image after applying thersholding, T is a threshold value,

Figure 2.4: Figure 2.2(a) after applying a thresholding operation: $I < 128$. Black represents 0 (false) and white represents 1 (true).



and $\square$ is some binary operation on real values (usually $>, <, \geq$, or $\leq$). Figure 2.4 shows the result of an example thresholding operation.

2.1.4 Binary Morphology

Building Blocks

Binary or black-and-white morphology is a collection of algorithms that transform the shape of a binary image. These algorithms are commonly applied after thresholding since thresholding converts a grayscale image into a binary image. Binary morphology algorithms generally are used to modify the shape or size of one or more connected regions. For example, there are morphology algorithims designed for growing or shrinking a region, finding a specific region, and filling gaps inside a region [6].

The two most basic binary morphology algorithims are dilation and erosion. Both operate on a binary image of any size and require a structuring element. The structuring element is a small binary image that acts as a probe. Examples of these two operations are shown in Figure 2.5

In dilation [6], the center of the structuring element is aligned over each pixel in the image that has a logical true value ("1"). Then, every pixel in the structuring element is logically OR'ed with the pixel directly below. The OR operation forces every pixel that is below a true pixel in the structuring element to be true in the image. Dilation only effects the edges of regions in the original image, effectively growing them by the shape of the structuring element.

Dilation is represented by the $\oplus$ symbol. It can be represented mathematically as the union of the set of true points in an image (A) and all points that the structuring element (B) covers when moved over the original set [6]:

$$A \oplus B \quad = \quad \bigcup_{b \in B} A_b \tag{2.5}$$

where A_b represents moving the structuring element b over a point in A.

Erosion [6], on the other hand, is an operation that effectively shrinks every connected region in the image. As with dilation, the structuring element is placed over every true pixel in the original image. However, here the probe detects cases when any true pixel in the structuring element is not

Figure 2.5: Morphological operations using a 3x3 structuring element of all 1s.

(a) Dilation (b) Erosion

over a true element in the image. If any such case exists, the center pixel is set to false, otherwise it remains true.

Erosion is represented by the $\ominus$ symhol. It can be represented mathematically as a subset of the true points in a binary image (A) in which the structuring element (B) is entirely contained in the original set [6]:

$$A \ominus B = \{z \in E | B_z \subseteq A\} \tag{2.6}$$

Along with dilation and erosion, a third, slightly more complex operation form the basic building blocks of most morophology algorithms. This operation is called the "hit-or-miss transform" [6]. It is fairly similar to erosion, but instead of only matching trues to trues, the hit-or-miss transform detects when every pixel in the structuring element (true or false) matches the element below it in the image. If a complete match is found, the pixel on which the structuring element is centered is set to true, otherwise it is set to false.

A common use for the hit-or-miss transform is to apply it multiple times to an image in order to thicken or thin regions. Using the proper structuring element (or a combination of multiple elements), the hit-or-miss transform can find boundaries in an image. This works because the pixels in a region that are not selected by the hit-or-miss transform are usually on the edge of that region. The boundary pixels then can be added to or subtracted from the original image in order to increase or decrease the size of all the regions in the image. In the case of growing the image by using the hit-or-miss transform, over successive iterations disconnected regions of the image will remain disconnected.

Combining Operations

These three basic algorithms can be used in succession to form more complex algorithms. Two commonly used algorithms are opening and closing [6]. Opening is performed by using the same structuring element to first apply a dilution and then an erosion. In closing, erosion is first performed followed by dilution. Symbolically [6]:

$$Opening : A \circ B = (A \ominus B) \oplus B \tag{2.7}$$

$$Closing : A \bullet B = (A \oplus B) \ominus B \tag{2.8}$$

After applying a closing operation, the resulting image will generally (depending on the choice of structuring element) contain the original region, but with small branches along the edges removed. Opening, instead of removing small edge effects, grows the region around them (generally) and also fills in holes inside a connected region.

2.1.5 Active Contours

Active contours or "snakes" is a method for finding desired contours or edges in an image [3]. This technique works by starting with a curve on the image ("the snake") and then iteratively moving that snake until it reaches a desired location. To use this method effectively, a good definition is required for what the snake is, how it will move, and what a desired location implies.

Setup

Snakes generally are designed to be closed curves, meaning that all paths around the snake lead back to their starting point (i.e., there are no end points). Later, it will be shown how to adapt closed curves methods to find open edges. We define a snake by both the location of all pixels on the curve and their order along the curve.

The snake moves in iterative steps determined by two influencing factors. The first is an internal tension or energy that resists separation between the points on the snake. This factor allows the curve to maintain its connectedness as it moves. The second factor is an external energy that is derived from the image.

The external energy is computed by applying a mathematical operation to the image that tends to have local minima at desired locations. In the case of edge finding, since edges are boundaries between regions, the snake is pushed towards areas with high gradients (the difference between neighboring pixels). This is accomplished by having large force vectors in low-gradient areas and small force vectors in high-gradient areas. Gradient descent then is used to "push" the snake along the energy field toward local minima. The iterative process is halted once the snake no longer moves.

Numerical Implementation

The important mathematical concepts required for implementing active contours are energy functions and difference methods. An energy function can be any image matrix operation or simply the image matrix itself. In this context with active contours, it is desirable to locate edges in the image. So, energy functions based on the gradient are frequently used.

Once the energy field has been computed and the snake initialized, difference methods are required in order to move the snake. Each pixel on the snake is "pushed" by two forces: the internal tension and the external image energy. To compute the net movement of the snake at each step, the internal and external vectors are added together.

To compute the internal force vector, either a first-order or second-order difference equation can be used. The difference equations are found by discritizing the partial difference equation for the snake [3]:

$$\alpha x_{ss} + \beta x_{ssss} + \frac{\partial E_{ext}}{\partial x} = 0 \tag{2.9}$$

$$\alpha y_{ss} + \beta y_{ssss} + \frac{\partial E_{ext}}{\partial y} = 0 \tag{2.10}$$

Here x_s represents the derivative of x along the snake curve, E_{ext} is the external energy of the snake, α is the first-order parameter, and β is the second-order parameter. Only the first-order case will be considered here (so we set β to 0). To compute the first-order difference vector, a tridiagonal banded matrix is used (second order would require a pentadiagonal banded matrix). The banded matrix is multipled by an ordered vector containing one dimension (the components of the resulting vector are computed separately) of the position values of the snake in order to determine the force on each point. The banded matrix [3] is shown here: (the constants a, b, c equal $2\alpha + 6\beta, -\alpha - 4\beta, \beta$, respectively.)

$$\begin{pmatrix} c & b & a & 0 & 0 & 0 & 0 & ... \\ b & c & b & a & 0 & 0 & 0 & ... \\ a & b & c & b & a & 0 & 0 & ... \\ 0 & a & b & c & b & a & 0 & ... \\ 0 & 0 & a & b & c & b & a & ... \end{pmatrix}$$

To compute the external force vector, we first use an energy function to define an energy map for the image. Two example energy functions are [3]:

$$E = |\nabla I(x, y)| \tag{2.11}$$

$$E = |\nabla(G_\sigma \star I(x, y))| \tag{2.12}$$

The first uses the image pixel values directly. In the second, Gaussian blurring, which uses a 2-dimensional Gaussian filter represented by G_σ, is applied to the image first. This function draws the snake to large gradients in the image, which generally correspond to edges between similar regions. The minus sign causes areas of high gradients to have the minimal values of the function. To improve the likelihood of the snake finding the correct gradient, a Gaussian blurring filter is often used before this step in order to smooth the image, eliminating the effects of outlying pixel values.

Once the energy map has been computed, a gradient descent algorithm is used to direct the snake toward local minima [3]. A step parameter is used to tune the magnitude of the external force on the snake, depending on desired results.

The discritized version of the snake partial differential equation (PDE) can be solved using an iterative equation set given here [3]:

$$x_t = (A + \gamma I)^{-1}(\gamma x_{t-1} - f_x(x_{t-1}, y_{t-1})) \tag{2.13}$$

$$y_t = (A + \gamma I)^{-1}(\gamma y_{t-1} - f_y(x_{t-1}, y_{t-1})) \tag{2.14}$$

Where A is the pentadiagonal banded matrix, I is the identity matrix, γ is the step parameter, and f_x and f_y are the partial derivatives of the energy functional.

2.2 Machine Learning Algorithms

Machine learning provides a collection of statistical methods for making predictions based on observations. There are generally two types of problems that machine learning attempts to solve: classification and regression [1]. Classification attempts to place observations in distinct collections, while regression attempts to predict a continuous, real value for each observation. The predicted values are known as *labels*. In this project, regression methods were utilized.

Data used with machine learning algorithms belong to one of two categories: training and testing. Training data are a set of observations for which the labels are already known by the algorithm. For testing data, the labels for the observations are unknown. A number of methods exist for splitting a data set into training and testing subsets.

2.2.1 General Principles

Regression Problems

To solve a regression problem, we must put the data set into a proper format. Each observation corresponds to one example from the dataset and includes a set of values known as features and a label (which may or may not be known). For the purposes of this project, it is assumed that all observations consist of the same features. For most algorithms, a feature matrix is formed from the data. Each row in the feature matrix corresponds to one observation and each column of the matrix corresponds to a specific feature. So, the (i, j) entry in the matrix corresponds to the value of the jth feature for the ith example.

For the training data, an $m \times n$ feature matrix has a corresponding $m \times 1$ label matrix that contains the label value for each observation. The goal of the algorithm is to predict the label matrix for the testing data, minimizing some cost function. For example, the most basic regression algorithm is linear regression where the cost function is least squares. The cost function for linear regression is [1]:

$$J = \arg\max_{w} ||\mathbf{y} - \Phi\mathbf{w}||^2 \tag{2.15}$$

Where J is the error, $\mathbf{y}$ is an $m \times 1$ matrix containing the row labels, Φ is the design matrix (where each row is one observation), and $\mathbf{w}$ is the parameter weight vector. The primal solution to this regression algorithm is found by computing the derivative of the error function and finding its minimum [1]:

$$w = (\Phi^T\Phi)^{-1}\Phi^T Y \tag{2.16}$$

Kernels

Often times in machine learning problems, it is desirable to create additional features for each observation by using a transforming function. This is done so that linear algorithms can be used to find non-linear relationships in the data. One example of this is apply polynomial operations to the data (e.g., $x^2, x^3, (x+1)^3$). When it is desirable to create many new features, the data set can become very large in the number of features and the solution then will be computationally expensive.

To simplify the process of transforming data sets, the "kernel trick" can be used [1]. This trick is applied by re-expressing machine learning algorithms in so-called dual form. In this form, the results only depend on the inner products between each observation in some space, so the original data set is not explicitly needed. The inner products do not necessarily have to come from the original observation space because a spatial transformation can be applied first. As a consequence, in the new form, instead of computing in the transformed space directly, we can use functions that implicitly join the two higher-dimensional, related data set using kernel functions. A kernel function acts on two observations from the data set and returns the dot product operation of the two observations in some high-dimensional space (Mercer's Theorem).

An example of the derivation of kernel form is shown in Section 2.2.2. Combining these two concepts (Mercer's Theorem and dual form) allows for transforming data into higher-dimensional spaces for regression, without going through the computationally expensive operation of explicitly computing the higher-dimensional feature matrix.

Two examples of kernel functions are the Gaussian kernel (k_1) and the polynomial kernel with degree d (k_2) [1]:

$$k_1(x_1, x_2) = e^{-|x_1 - x_2|} \tag{2.17}$$

$$k_2(x_1, x_2) = ({x_1}^T x_2 + 1)^d \tag{2.18}$$

Feature Selection using Wrapper

When solving machine learning problems, control can be exercised over the form of the regression solution beyond what is enforced by a specific algorithm's cost function. One instance in which it is desirable to add such control is for performing feature selection, which is desirable to counter over-fitting.

Over-fitting occurs when the statistical models in machine learning algorithms adapt to the specific features of a particular training set, which can introduce additional error in future predictions. A simple example of overfitting would be trying to predict Wednesday's temperature given that I only know the temperatures of Tuesday and Thursday. Using only this one trend to predict the temperature, the best guess (most probably) I can make would be $T_{Tuesday}/2 + T_{Thursday}/2$. While the temperatures on Tuesday and Thursday do have some impact on Wednesday's temperature, it is obvious that many other trends also affect the temperature. By only using one trend to make a prediction, we have overfit the data.

Over-fitting is guaranteed to occur in cases where the design matrix is rank deficient. In the case where there are more features than examples, such as is the case in this problem, the matrix is guaranteed to be rank deficient. To correct for over-fitting then, some technique must be used to limit the number of features used in the regression solution.

Feature selection fights over-fitting by limiting the number of features used by an algorithm. Wrapper [9] is one feature selection method that finds an optimal feature subset to use with a machine learning algorithm ('optimal' is defined in the context of the problem).

Wrapper works by iteratively searching the set of all feature subsets for an approximately optimal subset. The true optimal subset can be found, but the computation is extremely expensive (it requires exploring all possible subsets, e.g., when selecting 10 features from 100, the number of possible subsets is on the order of 10^{13}). One method for performing this search is using a forward greedy method. In this method, the wrapper selects a subset of features (hereafter referred to as the selected set) of features one-by-one. The selected set is initially empty. The wrapper starts by taking each feature of the data set as a feature matrix. It runs a machine learning algorithm with each of the feature matrices that has some validation scheme. From the validation scheme, an error is computed for each feature matrix. The feature that resulted in the smallest error then is added to the selected set.

In each following iteration, a similar procedure is applied to select one new feature. The difference is that the features in the feature matrices used with the algorithm consist of the selected set plus one remaining feature (i.e., not in the selected set).

Using this search method, the wrapper iterates until a desired error level is passed or a maximum number of features is reached.

With the above example (selecting 10 features from 100) the forward greedy method requires visiting around 10^3 possible subsets.

2.2.2 Algorithms

Four methods were used to solve the regression problem in this project: Regularized Linear Regression, Lasso Regularized Linear Regression (Lasso) [12], Support Vector Regression [13], and a more general technique called Wrapper [9]. Of these, Lasso and Support Vector Regression showed reasonable results on the data set.

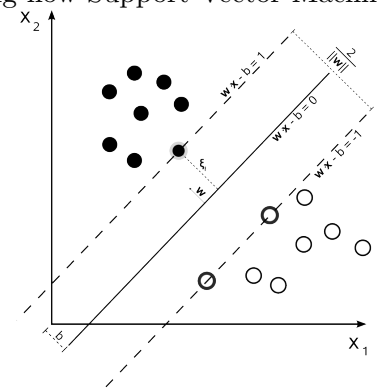
Lasso

Lasso uses an iterative method to compute the weight vector for the regression one feature at a time. At each iteration, Lasso selects the feature that is most correlated with the residuals found after using all previously selected features. It stops when a preset constraint on either the L1-norm of the weight vector or the total number of features is reached. In this manner, Lasso can produce an arbitrarily sparse regression solution [12].

Support Vector Regression

Support Vector Regression (SVR) is derived from the Support Vector Machine (SVM) algorithm, which is a member of the maximum margin family of classifiers. The goal of the SVM algorithm is to find a hyperplane that separates clusters of like data, given the constraint that the total margin, or distance between each point and the panel, is maximized. Due to this constraint, SVM solutions often are written only in terms of a subset of points in each cluster that are closest to the plane and thereby define its location without interaction of any other data (see Figure 2.6). These points are called "Support Vectors." This property makes SVM and related algorithms ideal for large data sets because they often are extremely efficient at making predictions.

Figure 2.6: Graph showing how Support Vector Machines lead to sparse solutions.



Mathematically, SVR (in primal form) attempts to find a collection of weight parameters w and a constant b such that the prediction error, given by the following function, is minimized [1]:

$$f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{2.19}$$

To find the solution, we can state the convex optimization problem [1]:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{2.20}$$

Subject to:

$$\begin{aligned} y_i - \mathbf{w}^T\mathbf{x}_i - b &\leq \epsilon + \xi_i && (2.21)\\ \mathbf{w}^T, \mathbf{x}_i + b - y_i &\leq \epsilon + \xi_i^* && (2.22)\\ \xi_i, \xi_i^* &\geq 0 && (2.23) \end{aligned}$$

Where ξ_i and ξ_i^* are Lagrangian multipliers representing the margin between the hyperplane and the observation points, C is a trade-off constant, and ϵ is used to create a "soft margin" that has some width instead of being a flat plane.

At this point, the optimization can be stated as a Lagrangian in primal form by introducing four new collections of Lagrangian multipliers: $\alpha, \alpha^*, \eta, \eta^*$ [13]:

$$\begin{aligned} L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ -C\sum_{i=1}^{l}(\eta_i\xi_i + \eta_i^*\xi_i^*) \\ -\sum_{i=1}^{l}\alpha_i(\epsilon + \xi_i - y_i + \mathbf{w}^T\mathbf{x}_i + b) \\ -\sum_{i=1}^{l}\alpha_i^*(\epsilon + \xi_i + y_i - \mathbf{w}^T\mathbf{x}_i - b) \end{aligned}$$

Next, partial derivatives of L can be taken with respect to the primal variables:

$$\begin{aligned} \frac{\partial L}{\partial b} &= \sum_{i=1}^{l}(\alpha_i^* - \alpha_i) = 0 \\ \frac{\partial L}{\partial w} &= w - \sum_{i=1}^{l}(\alpha_i^* - \alpha_i)x_i = 0 \\ \frac{\partial L}{\partial \xi_i^{(*)}} &= C - \alpha_i^{(*)} - \eta_i^{(*)} = 0 \end{aligned}$$

By substituting the constraints from the partial derivatives into the original, primal equation,

the optimization problem can be written in dual (kernel) form:

$$
\begin{aligned}
maximize\ & -\frac{1}{2}\sum_{i,j=1} l(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\langle x_i, x_j\rangle \\
& -\epsilon\sum_{i=1} l(\alpha_i + \alpha_i^*) + \sum_{i=1} l y_i(\alpha_i - \alpha_i^*) \\
& \text{subject to} \\
& \sum_{i=1} l(\alpha_i - \alpha_i^*) = 0 \ and\ \alpha_i, \alpha_i^* \in [0, C]
\end{aligned} \tag{2.24}
$$

From this form, a number of different methods exist to numerically find the SVR solution and solver libraries are available for many software platforms. As this Lagrangian is stated in dual form, it is easy to substitute a kernel function for the inner product [13].

Chapter 3

Process

3.1 Overview

Using image processing and machine learning techniques, a process was created that was able to accurately predict the relative DXA values using only the MRI data. This process will be an additional tool that the doctors can use for analyzing osteoarthritis in the knee.

The process begins by loading the MRI data into MATLAB from the DICOM format (DICOM is a standard file format for storing medical images). Certain data about each patient needs to then be associated with the MRIs: their relative DXA value, the range of slices that will be analyzed, and some data points from each slice that provide a rough indication of the region of interest. The first two pieces of information can be trivially stored in an associative array in memory. The last piece requires user input. To acquire this data, the slices from each patient's MRI data that will be included in the analysis are displayed to the user. The user is then prompted to select two points on the image: a point just to the left of center on the top edge of the tibia and a point about 100 pixels below the top edge on the right edge of the tibia.

Once the user input is received, the segmentation process works on each slice in a specific MRI set individually. Using the user-inputed locations, the image is cropped to a small region containing the top-right edge of the tibia. Filtering and clustering are then used to make an approximate identification of which pixels in the image are in a bone region and which are in a cartilage region. Using this approximation, an initial guess of the location of the top edge of the tibia is made. An active contours algorithm is then used to refine this initial guess. Finally, the region of interest is formed by taking a fixed length of the top edge and then selecting all pixels that are less than fixed distance below that curve.

After the regions of interest are found on all slices in a set, the spatial texture filters can be applied to each slice. Forty different filters were used in the analysis: eight Gabor filters and two variations on sixteen Laws texture masks. Each filter transforms a slice's region of interest into a texture map. Statistics are then taken on the values in the texture map across the regions of interest for every slice in a set. This results in five real values: mean, standard deviation, skewness, kurtosis, and variance. These five values are computed for each texture filter for each knee. The feature matrix is then formed using each knee as one observation and the texture map statistics as the features.

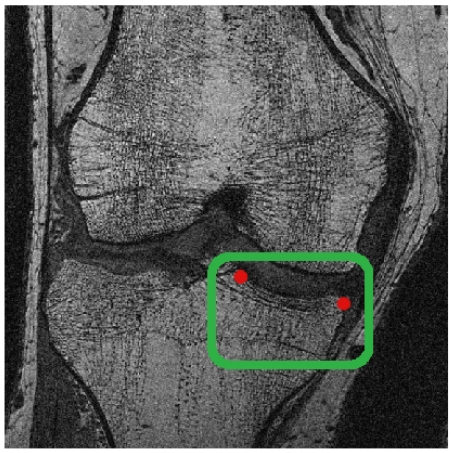
At this point, the computed data along with the relative DXA values are used in the regression algorithms. A number of different experiments were setup that used a variety of learning algorithms

and validation schemes. Ultimately, it was found that Support Vector Regression using a fourth degree polynomial performed the best. In a cross-validation scheme, the average root mean square error (RMSE) on the testing set was 13%.

3.2 Segmentation

A three-step process was developed to segment the ˜20 selected knee slices. First, the user selects a left and right boundary point for the region of interest, as shown in Figure 3.1. After selecting the two points, a rectangular region is cropped from the image (shown in green in the figure). This region is created by using the two selected points as references, and then expanding out from them by an arbitrary, fixed amount. Ideally, the user would only have to select the points on the first slice in each set and then the algorithm could adjust the points automatically as it progressed through the other slices. In practice, a more accurate segmentation was obtained by having the user select points on each image in the set individually. Using the selected points, the algorithm crops the image to a small region around the region of interest.

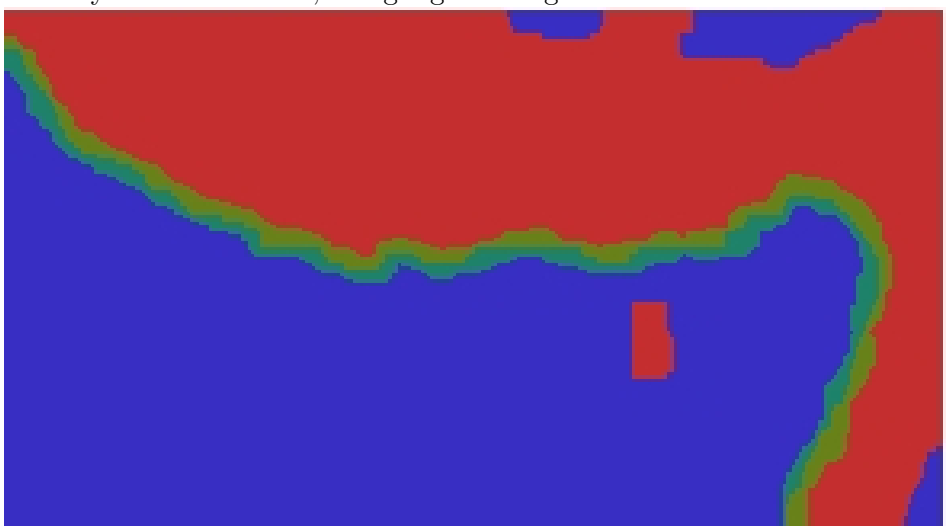
Figure 3.1: Step 1 of algorithm: user clicks two points on the image to initialize the segmentation. From the two points, a rectangular region is formed and cropped out of the image.



In the next step, the algorithm computes the local variance of a 15-pixel by 15-pixel neighborhood around each point in the cropped region [11]. The local variance value gives some information about the texture in the computational region and was found to be a strong discriminator between bone and cartilage. The local variance values are thresholded and then a few binary image morphological operations are applied to obtain an initial guess at the shape of the tibia. An example of this is shown in Figure 3.2. The top border of this shape was then selected as a first estimate of the boundary.

In the final stage, shown in Figure 3.3, the initial estimate for the border was used to initialize the active contours algorithm. This stage refines the curve that has been fitted to the edge. To adapt active contours methods to find this particular edge, several modifications had to be made to the general algorithm.

Figure 3.2: Step 2 of algorithm: a variance filter is used to determine an initial segmentation of the cropped region into bone (blue) and cartilage (red). The border between the two, which is the curve we ultimately want to extract, is highlighted in green.



First, since the border is not closed, a method for open snakes had to be developed. This was a simple change because it was possible to simply fix the endpoints of the initial curve and then remove them from any difference calculations in the active contours step. This method worked because the edge that the snake has to find is much longer than the edge that is ultimately used to create the region of interest. Therefore, small errors at the snake's endpoints do not affect the final results in the area of interest.

The second, problem-specific modification was to ignore the horizontal component of the external force vector. It was found that external horizontal corrections did not add to the quality of the segmentation. In fact, they decreased accuracy by causing points to cluster on top of each other.

An additional algorithm for manual selection also was developed for the approximately 25% of image sets for which the previously described algorithm did not obtain an accurate segmentation. This algorithm used a clustering technique (k-means) to segment the image into five regions of similarity [1]. The user then could manually select which regions corresponded to the tibia.

Once the boundary was found for each slice, it was trivial to select the region of interest by setting a fixed pixel width and height for the region and then using the computed border as the top of this region, as shown in Figure 3.4.

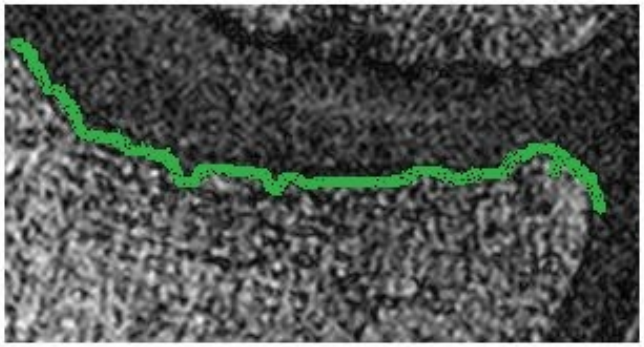
3.3 Texture Analysis

3.3.1 Selected Masks

A total of forty texture masks were used in this analysis (eight Gabor and two variations of sixteen Laws Masks). The choice of which texture masks was arbitrary, which is why a large number were used. The idea behind using so many texture features is that, if some of the texture masks provide usable information, feature selection (which occurs later) will be able to identify these masks.

The eight Gabor masks all had the same phase (0°), aspect ratio (1 : 1), and standard deviation

Figure 3.3: Step 3 of algorithm: the technique of active contours is applied to the border between the lower bone region and the cartilage in order to refine the location of the border.



of the Gaussian envelope (two pixels). The other two properties were varied to form eight combinations: wavelength (two pixels and four pixels) and orientation ($0°, 45°, 90°$, and $135°$). Two additional processing steps also were applied. First, the filter was cropped from eleven-by-eleven pixels (the size required to contain all the non-zero pixel values) to seven-by-seven pixels [7]. This does not change the response of the filter in a significant manner because most of the filter's energy is concentrated in the smaller region. It does, however, allow more data to be produced from the region of interest. Second, the filter was offset so that it would have a zero-mean. This step is important so that filter responds the same to dark and light regions in an image (i.e., intensity does not affect the filter response).

From the set of five one-dimensional Laws vectors, four were selected (spot, wave, edge, ripple) [10]. The level vector was not used because it is designed to respond to intensity rather than texture features. The four vectors were combined in all combinations using an outer product in order to form 16 five-by-five masks. An example mask is shown in Figure 3.5. An additional set of 16 texture maps was created by squaring the final texture values computed by the Laws masks.

3.3.2 Correction for Irregular Regions of Interests

Applying texture masks to the region of interest poised an interesting problem due to the irregular shape of the region of interest. Each region of interest was approximately 200 pixels wide and 40 pixels high, but the top and bottom edges followed the contour of the tibia as previously found using the segmentation procedure. This means that applying a texture mask to every pixel in the region of interest would result in values outside of the region being included in the analysis.

It was decided that no pixels outside of the regions of interest should be included in the texture analysis. To correct for this, a morphological erosion was applied to the region of interest by using a structuring element of all 1s, which was the same size as whatever texture mask would be computed. An example of this is shown in Figure 3.6. The resulting pixels would act as a list of all pixels in the region of interest that could be the center of the texture mask. Once the erosion operation was performed, a texture map was created by centering the texture mask on every pixel that was true

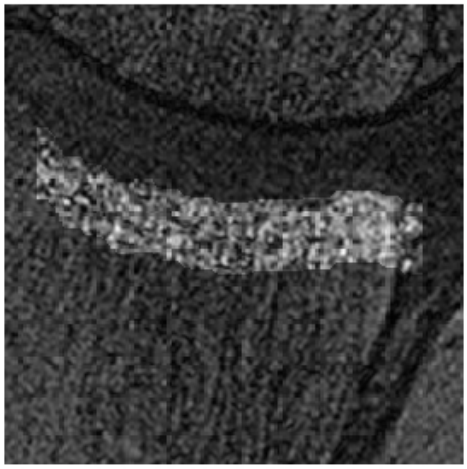


Figure 3.4: Example knee showing the region of interest that has been selected. Note the extension beyond the bone on the right. The texture analysis algorithm ignores the 20 right-most pixels in the region on every knee to compensate for this.

Figure 3.5: Example Laws mask created by combining the spot vector with the wave vector.

$$\begin{pmatrix} 1 & 0 & -2 & 0 & 1 \\ 2 & 0 & -4 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ -2 & 0 & 4 & 0 & -2 \\ -1 & 0 & 2 & 0 & -1 \end{pmatrix}$$

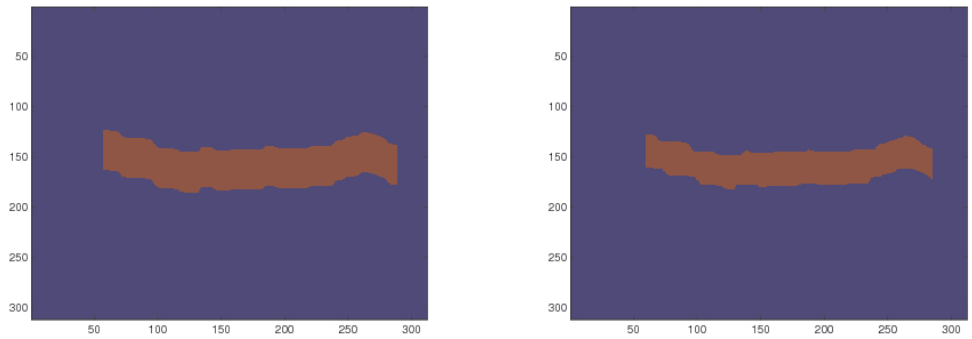
after the erosion. The resulting texture map was the same size and shape as the eroded region of interest.

All the values in the texture map for every slice in a particular knee were then collected into one array (essentially removing any spatial information).

3.3.3 Statistical Analysis

One data array was computed per knee that contains the texture map pixel values from every slice in that knee. Five statistics were then computed from each array (mean, standard deviation, kurtosis, entropy, skewness). The choice to use these statistics was taken from previous literature [4]. These five statistics provide concise information that describes the texture space on a particular knee for a particular filter.

Figure 3.6: Erosion of an example region of interest to show which pixels will be used in texture analysis.



(a) Original ROI

(b) ROI eroded by a seven-by-seven pixel structuring element for the application of a Gabor filter

3.4 Regression Analysis

At this stage, data had been collected for 42 knees. Using the texture data for each knee to form a feature matrix and the relative DXA values as the label, a variety of regression experiments were run. The goal was to find a regression model that would have low error in predicting relative DXA on a validation set, using only a small subset of all possible features. To accomplish this feature selection task, the wrapper method was used.

3.4.1 Initial Experiments

Initially, a variety of learning algorithms was tested using a random 70-30 split (70% of the data was used in the training set, 30% for the testing set) [1]. The two algorithms that showed the best initial results were Lasso and kernel Support Vector Regression. Both were able to achieve an RMSE of under 20% on the testing set. (All RMSE throughout this analysis are given in terms of % RMSE. This is calculated by taking the RMSE error and dividing by the range of values.) Due to its more flexible and more minimalistic nature, only kSVR was used for further refinements.

3.4.2 Further Refinements

Further refinements were made on the initial results in an attempt to improve precision. In this stage, a variety of kernels was tested with the kSVR algorithm.

The first refinement was to use cross-validation instead of a random split. Four-fold cross-validation was chosen, so all testing sets contained ten observations. From the cross-validation results, the average RMSE for the four disjoint testing sets was computed. This metric was used with wrapper to refine feature selection. To accomplish this, wrapper was run iteratively. Based on this analysis, it was found that the fourth-degree polynomial kernel had the best performance. On the entire data set, an RMSE error of 15% was achieved (with 15 features).

The final refinement was to inspect the predicted relative DXA values for each knee. The five knees that had the least-accurate predictions were removed entirely from the data set and

the regression analysis was re-computed. The five removed knees all exhibited substantially worse performance than the remainder of the data set. By removing these knees from the analysis entirely, a final RMSE of 13% was achieved (with 15 features selected).

In the knees that had the worst performance in this analysis, a few anomalies were found. One of the most prevalent was the presence of large dark regions in the image. These regions may have a medical explanation, so manual review by doctor should be used as a first pass on all images with this anomaly. Some of the images also showed low image quality or distortions caused in the process of acquiring the image. To account for this, MRI calibration data could also be included in the analysis or a researcher can manually remove problematic images.

3.5 Results

From the original diverse bank of filters that were included in the experiment, no expectations about which filters would be most present in the solution was made. A wide range of filters was used so that the learning algorithm would have many features to choose from.

Table 3.1 lists the features that were selected by the wrapper algorithm, indicating that they have the strongest correlation with the relative DXA label. This implies that, based on our test data and algorithm choice, that these 15 texture filters have the strongest discriminatory power for predicting the label value out of all of the original 120 filters.

From the perspective of our intuition, this implies that the texture pattern encapsulated in each of the selected filters should have some relation with the texture in the MRI image and thus what that image signal represents: the underlying trabeculaer morphometry of the bone.

Based only on the data found in Table 3.1, developing an intuitive sense of what the texture patterns in the selected filters are would be difficult for a variety of reasons. The primary concern is that the texture filters are small (5x5 or 7x7), discrete signals derived from analog band-pass and band-reject filters as described in Section 2.1.3. In order to base our intuition on the textures generating waveforms, we would need to account for strong sampling and quantization effects that are introduced by digitizing the signals.

Further, it must also be taken into account that the learning algorithm was not trained on the raw output of the texture filters, but on statistics of their distribution over the ROI. By design, these statistics provide information on the distribution of a texture pattern in a region, not just the texture pattern itself. Since this data was computed and summarized using learning algorithms, which find the mathematical, not intuitive, structure of the data, it would require further introspection and controlled testing to determine what specific changes in the texture pattern distribution accounted for the discriminatory power of a particular filter.

One way to develop a better intuitive feel of the data would be to perform a simulation, in which an idealized bone could be used and manipulated in known ways. The simulation could be used to isolate variables in the trabeculaer morphometry and determine how these variables cause changes in the final output of the texture feature. This would be costly in terms of setup time and computation time, would provide insight into how the texture filters work in reality, but would probably be of little value to the medical research community. A simpler simulation on generalized texture structures would give some insight, but may not be applicable to this particular problem.

Despite these shortcomings, it is possible to make some speculation on what factors are important in these results. There are a few possible patterns in the results.

First, in the 15 selected Gabor filters, four of the filters had two-pixel wavelengths and orienta-

Table 3.1: The 14 features selected in refined algorithm

Filter Type	Parameter 1	Parameter 2	Statistic
Gabor	Wavelength: 2px	Orientation: $\pi/4$	Skewness
Gabor	Wavelength: 2px	Orientation: $\pi/4$	Entropy
Gabor	Wavelength: 2px	Orientation: $3\pi/4$	Skewness
Gabor	Wavelength: 2px	Orientation: $3\pi/4$	Entropy
Gabor	Wavelength: 4px	Orientation: $\pi/4$	Standard Deviation
Gabor	Wavelength: 4px	Orientation: $\pi/2$	Standard Deviation
Laws	Vertical: Edge	Horizontal: Edge	Entropy
Laws	Vertical: Spot	Horizontal: Edge	Mean
Laws	Vertical: Wave	Horizontal: Wave	Entropy
Laws	Vertical: Ripple	Horizontal: Edge	Standard Deviation
Laws	Vertical: Ripple	Horizontal: Wave	Standard Deviation
Laws	Vertical: Ripple	Horizontal: Wave	Kurtosis
Laws	Vertical: Ripple	Horizontal: Ripple	Standard Deviation
Laws	Vertical: Ripple	Horizontal: Ripple	Kurtosis

tions of 45° and 135°. A wavelength of two-pixels implies that every pixel is a peak in the intensity pattern. This suggests an abundance of small changes in the images texture in a diagonal direction. Skewness and entropy were the two statistics used in these four selections. Skewness suggests that the presence or absence of this texture pattern in part of the image is an important discriminating factor. Entropy suggests that randomness in where this texture is located is the important factor.

Also, in the selected Laws masks, five masks had either a horizontal or vertical Ripple component or both. Ripple is derived from a band-reject filter, so it looks for high or low frequencies in the texture. Since it detects high frequencies, it may look for pattern similar to that of the two-pixels wavelength Gabor filters. However, in this case, the two important statistics were kurtosis and standard deviation. Both of these statistics provide information about the concentration of the distribution.

Overall, from speculation on these results, we would intuitively expect that the presence or absence of high-frequency texture and how this texture is distributed throughout the ROI is the best discriminator for predicting relative DXA. This is consistent with the problem as the trabeculae are often smaller than one pixel at the resolution of the MRI image data.

Chapter 4

Conclusion

4.1 Summary

By combining all the individual components into one process, a system has been created that takes an MRI data set and some additional medical data from the user and outputs a model for predicting relative DXA values. The prediction accuracy achieved in the experiments validated the utility of applying image processing and machine learning concepts in a medical application.

Specifically, the results from the analysis in this report provide a new means of analyzing how changes in the trabeculaer morphometry of bone participate in the progression of OA. Medical researchers are just starting to understand this process and currently do not have reliable means for tracking these changes and using them to diagnose OA.

The current standard, relative paBMD scores, which served as the benchmark for validating this research, is currently regarded as experimental and inconsistent in the medical community. In this research, texture-analysis tools were used as an accurate predictor (on the sample set, a RMSE of 13% was achieved) for the existing standard. Because of this, these tools may eventual lead to new standard that replaces the current one.

The strong correlation that was found between the texture data and the relative paBMD scores suggests that a previously unknown relationship between observed bone mineral denisty and specific changes in trabeculae morphometry may exist. Since these texture methods were able to accurately predict a current standard, the possiblity exists that these methods may eventually become a stronger diagnostic standard than any that is currently used.

Eventually, the research methods explored here could evolve into new MRI diagnostic tools for OA in perarticular bone. These tools could aid doctors in the early diagnosis and progression tracking of OA.

4.2 Extensions and Further Work

In assembling and reviewing the final process, a number of points for futher work and additional features were identified. Some suggestions are based on revising and/or validating assumptions that were made in the development of the current process. Others involve additional work that would be desirable from a usability and/or performance perspective.

4.2.1 Automation

The original goal of the project was to create an automated or minimally interactive algorithm. This would accomplish two goals. First, it would make the system as easy to use as possible for the medical practictioners. Second, a robust framework that requires minimal user input has a lower chance of operator error occurring.

The solution presented here requires two clicks per MRI slice at the initialization stage. For an experienced user, this takes around 60 seconds per knee to complete. This step could be reduced by creating a refitting algorithm that only requires the user to select points on one slice and then extrapolates to find the same points on the remainder of the set.

Such an algorithm initially was included in the segmentation process, but it was found to have limited accuracy and was unstable on a large portion of the data set. As a consequence of this, it was replaced by the more manual procedure.

An additional possibility that was not explored is implementating a registration algorithm that can fit any knee in the dataset to a generalized model. Theoretical work exists in previous literature [6] that describes this sort of algorithm and some implementations have been used for other applications.

4.2.2 Performance

All executions of this algorithm were performed on a Toshiba laptop running Ubuntu 9.10 Linux with a 2GHz dual-core Intel Centrino Duo processor and 3GB of RAM. To execute the entire system in MATLAB, including user input, takes, on average, approximately 15 minutes per knee with a range of 12 minutes to 20 minutes. This range is caused by the rate of convergence of the iterative algorithms used in the process.

Several areas for possible performance increases have been identified, though these are largely outside the scope of this investigation. The two primary suggestions are refining the matrix algorithms to be optimized for this particular algorithm and including support for parallel processing.

4.2.3 Segmentation

The segmentation algorithm could be improved by exploring several areas.

One idea is to use a macrosegmentation that focuses on the whole image rather than the cropped image. One way to do this is to shrink the image initially and then perform a clustering segmentation to identify all the bone in the image efficiently. From this point, the correct region could be identified on the original image and a refinement algorithm, like active contours, could be used to finalize the segmentation.

Numerous other approaches exist that could potentially yield higher-quality, faster segmentations that would work on a larger portion of the data set. One idea that was explored during the course of this project was to use Principal Component Analysis (PCA) on a vectorized list of pixels in the image. Initially results were promising, but this direction was not pursued further.

4.2.4 Texture Analysis

As the texture filters used in this process were chosen in an arbitrary manner (though based on previous literature), additional improvements could be made by testing additional texture filters and using a larger variety of parameters with the Gabor filters. The final results of this project

validated the usefulness of spatial filtering techniques, so additional exploration could lead to the discovery of more useful tools. The biggest limitation on the number of filters that could be tested is the size of the data set. A larger data set would be needed to test additional filters.

Another technique that was explored over the course of this project, but not included in the final analysis, was a concept called Gray Level Co-occurrence Matrix. This method provides a way of quantifying how much a pixel differs from a neighboring pixel some distance away. Taking statistics on the co-occurrence matrix provides another means of generating numerical data for texture.

Additionally, the statistical methods used to summarize the texture data were fairly simple. Other parameters could be included (beyond mean, standard deviation, kurtosis, skewness, and entropy) in the analysis. Further, additional methods of creating feature vectors from the texture data could be explored.

4.2.5 Data Analysis

A variety of machine learning algorithms were used to analyze the data generated from the texture analysis. As library implementations are readily available for many learning algorithms, more regression methods could be easily explored.

Since the algorithms used in this project were effective, the primary remaining challenge is to further understand the results. The combination of using spatial filters and machine learning algorithms implies that the final predictive model is based around unintuitive features. From what has been explored so far, it is unclear why certain texture features are predictive. Determining the correspondence between the mathematics and physical forms could lead to additional understanding of the medical issues with which this research is concerned.

Bibliography

[1] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. 2006.

[2] G. Lo et. al. Strong association of mri meniscal derangement and bone marrow lesions in knee osteoarthritis: data from the osteoarthritis initiative. 2007.

[3] M. Kass et. al. Snakes: Active contour models. 1987.

[4] M. Sharma et. al. Evaluation of texture methods for image analysis. 2001.

[5] R. Bannuru et. al. Therapeutic trajectory of hyaluronic acid versus corticosteroids in the treatment of knee osteoarthritis: A systematic review and meta-analysis. 2009.

[6] R. Gonzalez and R. Woods. *Digital Image Processing*. 1992.

[7] HHT. Gabor filters for texture analysis. 2002.

[8] A. Jain and F. Farrokhnia. Unsupervised texture segmentation using gabor filters. 1990.

[9] Mike Korpela. Introduction to variable selection; part 2: Wrappers for feature subset selection. 2006.

[10] KI. Laws. Textured image segmentation. 1980.

[11] JS. Lee. Digital image enhancement and noise filtering by use of local statistic. 1980.

[12] Karl Sjostrand. Matlab implementation of lasso, lars, the elastic net and spca. 2005.

[13] Alex Smola and Bernard Schlolkopf. A tutorial on support vector regression. 2003.