7-19-1996

MRI On the Fly: Accelerating MRI Imaging Using LDA Classification with LDB Feature Extraction

Y Joy Ko
Dartmouth College

Michael B. Taylor
Dartmouth College

MRI On The Fly: Accelerating MRI Imaging Using LDA Classification with LDB Feature Extraction *

Joy Ko and Michael Taylor
Departments of Mathematics and Computer Science
Dartmouth College
Hanover, NH 03755

July 19, 1996

*Dartmouth College Department of Computer Science Technical Report PCS-TR96-290. Funding for this project was provided by the Class of 1939 Office of Residential Life Senior Scholars Program and the Richter Grant Program

Acknowledgments

I, Joy Ko, wish to acknowledge, with deep gratitude, my ever-growing family. Their support proved invaluable and their curiosity in the development and execution of this thesis sustained my own. I wish also to thank my partner, Michael Taylor, with whom it has been my privilege and pleasure to work. As partner, instructor, team-player, he has admirably demonstrated in all roles that patience, indeed, is key.

I, Michael Taylor, would like to thank my mother, Mary-Lynne, who has supported me selflessly from the beginning of time. I am grateful to Rodney Page, my high school computer science instructor, for his amazing enthusiasm and encouragement. I thank Jeff Swartz for helping to develop my abilities and for his excellent ability to flatter. Scott Silver requires significant recognition because his advice and experience (both in and out of class) have been a much appreciated resource. Finally, I thank my partner, Joy Ko, who never ceases to impress me. Working and learning with her has been one of the great highlights of my Dartmouth experience.

Additionally, we would like to thank Geoff Davis, Sumit Chawla and Doug Warner, who proved great resources in the development of our thesis. Finally, we would like to thank Dr. Dennis Healy, our thesis advisor. His perspective, guidance and experience were extremely valuable; his patience during our temper tantrums was admirable.

This joint thesis is but one result of a five term partnership that began in the summer of 1994. We have progressed from working within the confines of assignments given in the classroom to the free and internally motivated area of research.

By necessity, our topic is situated along the boundary of computer science and mathematics, a position that enabled each of us to lend unique (and often exclusive) expertise to the problems involved. This precarious position proved particularly valuable because it forced us out of our comfort zones into fields in which we had little experience. As a result, this thesis was a continuous learning experience. Such an arrangement required that we work effectively and closely together. This was not always an easy task, but we think that this thesis is a good indication of our success.

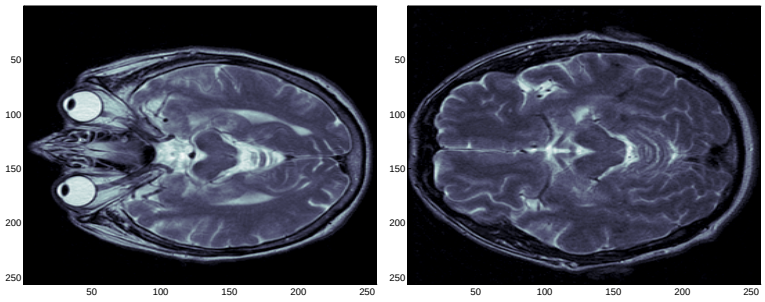


Figure 1: Trans-axial head section images

Introduction

Magnetic Resonance Imaging, or MRI, is a non-invasive clinical diagnostic tool. The speedier alternative, X-ray computed tomographic imaging, gains its contrast through measurements based on changes in density and is thus impractical for the examination of soft tissue which has near-constant density. MRI, on the other hand, provides excellent contrast when applied to soft tissue and has become indispensable in clinical medicine. Figure 1 shows two MRI images corresponding to a particular trans-axial head section.

MRI image acquisition is unfortunately very slow, which can make a routine examination a costly and inconvenient affair. For decipherable results, a patient is required to lie still for the length of the examination which can range from seconds to an uncomfortable number of minutes. Slow imaging prevents MRI from being a practical tool in the emergency room where a patient may be unable to lie still. Additionally, MRI is currently unusable in ambitious areas of study such as heart imaging and joint and muscle motion studies where the focus of study is very mobile. Reducing MRI imaging time will improve an already prominent medical tool and enable exploration of its exciting potential. This is the focus of our paper.

1 Problem Formulation

To reduce the effects of any bottleneck, one can either make the bottleneck larger or decrease the amount passing through. In the case of MRI, the bottleneck is the amount of time needed for image acquisition, a process that involves a series of measurements that pinpoint different areas of the desired image. The problem is compounded by the fact that numerous images are usually acquired during an examination. Decreasing the acquisition time has traditionally involved improving the hardware at great expense or scheduling orthogonal acquisition requests to allow for proton spin cycle latency. The alternative, decreasing the total number of measurements required, becomes quite attractive given that scheduling improvements have already been quite thoroughly explored and that it requires few, if any, modifications to existing hardware designs.

The question then becomes, how do we reduce the number of measurements required? Certainly the well-seasoned practitioner could improve this figure by reducing the number of requests for images, taking only as many images as necessary by anticipating which areas of the patient to examine. If successful, this certainly does reduce the total number of measurements needed. We

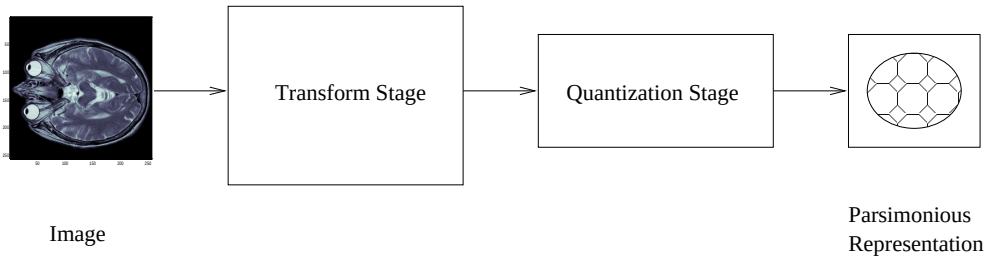


Figure 2: The two-stage process of lossy compression

take a different approach and investigate the possibility of reducing the number of measurements required for *one* image. This approach is inspired by the field of compression where the central problem is how to represent data most compactly in order to transport it over some limited bandwidth medium. In this case, we can view the MRI device itself as the "limited bandwidth medium." On first glance, the compression problem seems rather unrelated to the MRI problem. After all, we do not have the data yet in any computer form, so how can we use an algorithm to "package" it? The solution becomes more easily apparent after closer examination of compression which is the main emphasis of the next section.

2 Compression Overview

Compression methodology is easily divided into two classes: lossless and lossy. *Lossless compression* has the property that the decompressed data is exactly the same as the precompressed specimen. It trades short representations for more probable datasets against longer representations of less probable datasets. *Lossy compression* has the property that the decompressed data is not necessarily identical to precompressed data due to the omission of some information. This may seem disturbing on the onset but the essential idea is that the information that is discarded has been determined to be extraneous. The final representation of the data still contains the crucial information. For instance, suppose we want to pick a card from a poker deck and write a message to someone telling them what card we picked. We *could* send the entire image but clearly a more efficient representation is simply the text "Queen of Spades" etc. Observe that lossy compression must be tailored very specifically to the input and output data sets in order for the results to be intelligible. "Queen of Spades" may not be the appropriate representation for your resume, for instance.

In a similar manner, there are many lossy compression methods intended expressly for the compression of images. Typically, the lossy part of compression is described as a two stage process, as shown in figure 2. The first stage is called the decorrelating stage, or the *transform* stage and the second stage is called the *quantization stage*. The purpose of the transform stage is to come up with a new representation of the images such that the relevant information is packed into a relatively small number of axes. The actual compression comes in the quantization stage. During this stage, a certain quantity of bits is allocated to each axis, presumably with the objective of reducing the total bit usage. The idea here is to maximize the signal to noise ratio; in other words, to minimize the amount of storage allocated to axes which have little overall effect on the appearance of the image. The ideal scenario in light of compression is that the variance of the

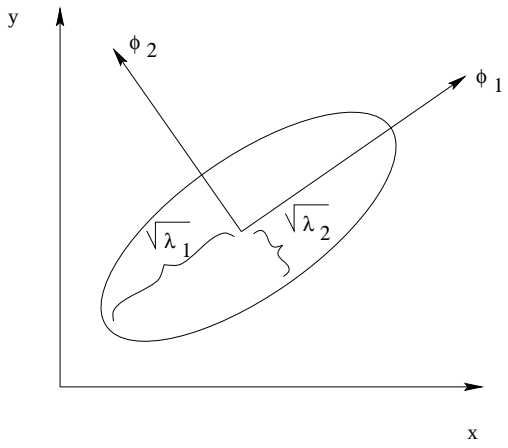


Figure 3: A 2-d illustration of the K-L transform

distribution is only significant in the direction of a small number of basis functions chosen in the transform stage. In this way, those basis functions whose coefficients have small contribution can be truncated without degrading the reconstruction of the images. We will describe in detail below some common transform methods.

2.1 Finding an Optimal Basis for a Class of Images

A key concept in the transform stage of lossy compression is the idea that an efficient representation of the images can be found using the fact that the images in each class are related to each other in some way. For instance, image transforms typically attempt to take advantage of the smoothness property of an image–that is, the tendency for adjacent pixels to be similarly valued. One approach to finding such a representation is given by the Karhunen-Loève (K-L) decomposition, also known as the method of principal components.

The K-L decomposition seeks to represent the class of images using a smaller set of features that takes advantage of the similarity between images of the same class. Mathematically, the K-L decomposition gives an expansion of a random process with finite second moments in a special basis of orthonormal functions. The expansion coefficients are random variables obtained as inner products of the process with the basis elements. The basis is chosen so that the expansion coefficients are uncorrelated random variables. Figure 3 illustrates this process in two-dimensions.

Let us represent the distribution encircled by the ellipse using the coefficients associated to the basis $\{(1,0),(0,1)\}$. Suppose we want a more compact representation and choose to represent this distribution using only the coefficient corresponding to one of the basis vectors. The *expected error of truncation* would be large since the information contained in each basis function is about the same. Large expected error of truncation indicates that this representation, although certainly more compact, is probably not sufficient to adequately distinguish one image in the distribution from another. Suppose, on the other hand, that we choose to represent the distribution using the basis $\{\phi_1,\phi_2\}$, where ϕ_1 and ϕ_2 are the eigenvectors of the covariance matrix of the distribution. The corresponding variances of the distribution along these principal axes are λ_1 and λ_2. We can see that more information is contained in the coefficient associated to the basis function ϕ_1 and if λ_2

is small enough, then the distribution might very well be approximated using only the information in the ϕ_1 direction.

The K-L basis is characterized by many attractive properties. The most useful one for the purposes of compression is that the decay rate of the K-L coefficients is rapid, which in turn corresponds to small expected error of truncation. For our purposes, this is precisely what we want: having a representation that concentrates the information in a relatively small number of coefficients so that considerable omission of coefficients can be done without significantly degrading the reconstructed image. The actual compression comes in omitting the higher order information, which occurs in the quantization stage. For *gaussian* distributions, distributions that are characterized completely by its second moments, the K-L basis is optimal.

The drawback of the K-L decomposition is that it involves the diagonalization of the covariance matrix whose dimension is that of the initial feature space. In the case of MRI images, the dimension of the initial feature space is 256×256. Since diagonalization is an $O(N^3)$ operation, this is not a very practical insertion into an algorithm whose primary consideration is time.

Fortunately, we can use Coifman and Wickerhauser's Best Basis Algorithm that approximates the K-L transform by restricting the optimization problem to a specially chosen library of bases, such as Wavelet Packets. The basis that is chosen is tailored for the class of data in mind and although a sub-optimal solution, gives comparable compression results to the global optimum. Before we can describe the best basis algorithm, we will supply some detail about the standard transforms that are the building blocks for this algorithm.

2.2 Fourier and Wavelet Transforms

One transform used extensively in signal processing is the Fourier transform. The Fourier basis is used frequently in sound analysis to break a signal down into its harmonic frequencies. This is an effective representation except that it does not give any information about when the component frequencies occurs in the signal. This is akin to having a graphic equalizer on a stereo which updates once for every song. If we examine each second of the signal, then we can effectively build a map to describe which frequencies occur at what time. This is essentially what is called a *time-frequency map*, which presents a way of representing both signal content and local signal features. In an image, such a map is useful because we are often concerned with the variation in spatial harmonic content throughout the image. [1] Note however that higher frequencies can be divided into much thinner slices of time while lower frequencies require more time to determine. Thus an enhanced dynamic tiling of space and frequency can be arranged which gives better support and definition for high frequency content. Figure 4 juxtaposes the time frequency constant tiling for the windowed Fourier decomposition and the dynamic tiling in wavelet transform. A large collection of such tilings is provided by Wavelet Packets bases, which provide us with a library of bases with which we can represent an image. This library enables us to customize the set of bases to the individual image or class of images. Clearly, there are as many possibilities of bases as possible tilings of the time-frequency plane. Although large, this fairly comprehensive library offers a manageable collection of bases. The best basis algorithm of Coifman and Wickerhauser finds the best basis of this form with respect to an appropriate cost function from such a library of bases. [4]

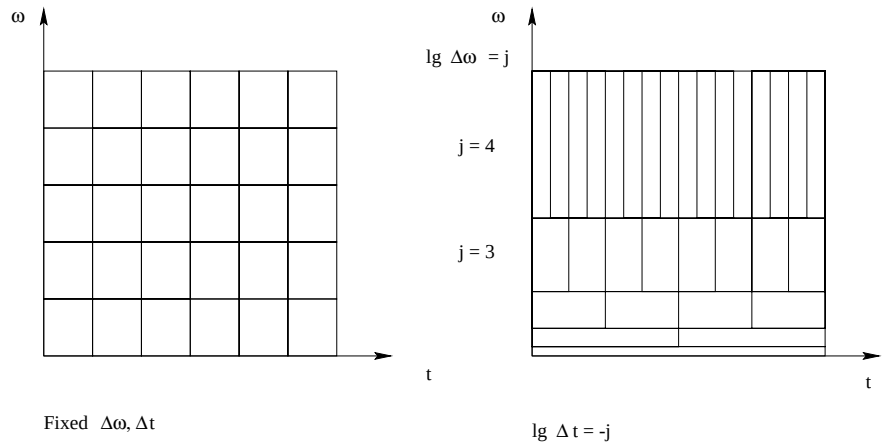


Figure 4: Time-frequency tiling for windowed Fourier and Wavelet Packet transforms

2.3 Best Basis Algorithm

Before we can begin to discuss an efficient manner to find the best basis, we need to to define what is meant by 'best'. Recall that the K-L basis is optimal because it concentrates most of the relevant information about a signal into a relatively small number of features. An appropriate cost function for this algorithm should similarly measure a signal's concentration of information in a given basis. A natural choice for this is the *Shannon entropy* function. Let $\mathbf{p} = \{p_i\}_{i=1}^n$ be a sequence which can be viewed as the normalized energy distribution of the signal. The entropy cost of this signal is given by

$$E(\mathbf{p}) \stackrel{\triangle}{=} -\sum_{i=1}^{n} p_i \log_2 p_i.$$

For a library of bases, the best basis for a signal is the one for which this information cost is minimized.

The first step in this algorithm is to create a *wavelet packet table* for a given signal, which can be seen as the decomposition of the signal into low and high frequency components at increasingly finer scales. At each step of the decomposition, a signal representation from the previous step is subdivided into low and high frequency components, by the "scaling" and "wavelet" functions. The end result is a packet table of depth $\log_2 N$, where N is the length of the original signal; each level j contains the 2^j subdivisions of the signal. This wavelet packet table is over-complete, and contains all the possible bases considered in the library. For illustrative purposes, we can view this packet table as a binary tree where each complete subtree is a complete representation of the signal. Observe that each node in the tree represents a certain portion of the signal which is exactly the portion represented by its children. Associated with each node is a value that reflects the cost of expanding the signal in a basis that includes the node. To find the best basis in such a library, we start at the bottom of the tree and compute the cost of each node. We also compute the cost of each node in the level right above. For each of the parent nodes on the level above, we can compare its cost with the sum of its children's costs. We keep the representation that has the smaller cost function. This continues until we reach the root of the tree.

The best basis algorithm can be extended to n-dimensional images very easily. Each node of the tree instead has 2^n children. This algorithm can also be extended to find the best basis for a class of images, rather than for a single image. The wavelet packet table for the entire class is created by finding the packet tables for each image in the class and then summing together all the terms in the same position within the table. Each term in this aggregate table is then normalized by the sum of the magnitudes of each image in the class. The best basis algorithm can then be applied to this class wavelet packet table, yielding an overall best basis for the class of images. Note that there may be a better choice of basis for each individual image.

2.4 MRI and Compression

In the case of MRI, a lossless algorithm is not a reasonable way to reduce the number of measurements for an image acquisition. The only way to gain a complete, lossless representation of the image would be to acquire the entire image, which is exactly the task that we want to avoid. A lossy algorithm holds more promise since it allows us to leave out measurements that may not be so important for a particular diagnosis. MRI has the unique property that images can be acquired in a variety of bases. [1] Thus, we can perhaps find a basis which is tailored to MRI images from a particular area of the body. Given our recent exploration into the Best Basis algorithm, we can find such a basis. For future acquisitions in that area of the body, we can acquire the image in this basis, omitting any basis functions with little or no information content. In this way, we have eliminated a significant number of measurements while also retaining image quality.

3 Classification To Improve Compression

Up to now, we have focused primarily on finding an optimal basis for images on the assumption that the images within a group have a gaussian distribution. After all, it is for gaussian distributions that the approximate K-L decomposition, or best basis, has proven to be an efficient means to compactly represent a distribution of images. Unfortunately, for *non-gaussian* distributions–distributions which fail to be adequately characterized by second moments–the effectiveness of K-L is also weakened. What we really want is to expand the notion of optimality across an arbitrary *collection* of images, rather than rely on the oversimplistic assumption that the space of possible images falls in a gaussian distribution.

Specific to our application of MRI imaging, we would like to take advantage of the similarities between images without losing the important information unique to each image. The MRI images that we work with are typically arrays of 256×256 pixels whose grayscale values represent a weighted density of the hydrogen in water and fats of tissues in a planar slice of the subject. [1] We can imagine that in 256×256-dimensional space, each image is represented by a point. In a very simplified example, we can observe that given a set of trans-axial head section images, some images might have eyes and others without. This might translate into two distinct clouds in 256×256-dimensional space. If we separate the set of images into these two classes, then the bases that we come up with to represent the images in each class are more specific to the image itself and of smaller dimension. The gain in predicting the possible classes that an image might fall in is that the representation of that image within the specific class that it is assigned to is both efficient and more representative of the image itself.

This motivates the use of *classification* to improve compression over many images. We introduce the idea of a *gaussian mixture model* which is a partitioning of the estimated distribution of all possible images into classes. The construction of the classes is done to minimize the expected error of truncation which in turn determines how effectively the images can be compressed. Classification is a two pronged process:

1. Finding a good way to choose the classes for a set of images

2. Deciding on a classifier that assigns a given image to a class

3.1 Determining Classes

An important concept in data analysis is the notion of using past experience to model the space of possible images. In this first step of classification, we begin with a set of images that we have already acquired which will be called a *training set*. The goal is to find the best partitioning of the training set into classes in the hopes that the behavior of the training set is a just representation of the entire image space.

To quantify what is meant by the best partitioning, we must come up with an adequate estimate of coder performance. For a given class, a rough estimate of coder performance is the volume of the class, which is given by the product of the variances of the distribution along the basis functions used to represent the class. This volume has shown to be a good predictor of coder performance, with small volume indicating better performance. In light of the entire space of images, we want the total volume over all the classes to be minimized. The total volume is given by the sum of the volumes of each class weighted by the number of images in each class. Suppose we have already specified the number of classes, k, that we want to split the training set into. Let N represent the total number of images in the training set and N_i the number of images in class i. The explicit formula of the total volume of a partitioning is given by

$$TotalVolume = \frac{1}{N}\sum_{i=1}^{k} N_i V_i,$$

where V_i is the volume of class i.

The algorithm that we use to create the desired gaussian mixture model is the k-bases algorithm which is an iterative process that produces a partitioning of the training set into k classes based on the chosen optimality criterion. The advantages of this algorithm are that it is easily adaptable to any criterion and that it produces disjoint classes. The algorithm proceeds in the following fashion:

Step 1 We begin with an initial partitioning of the training set into k classes. The k is a predetermined value. To get an initial partition, we can simply choose k random points as the initial centroids of the classes. We then assign each image in the training set to the class corresponding to the centroid that is closest to it.

Step 2 Compute the centroid and the approximate K-L basis for each class.

Step 3 Let x denote an image in one of the classes. We swap x to every other class, calculating the total volume of the new partition at each swap. The class that we switch x to is that which will minimize the total volume.

Step 4 Go back to step 2 until we have reached until the total volume no longer changes upon swapping an image. This is bound to be reached since each iteration of the k-bases algorithm decreases the total volume. Moreover, the total volume is bound below by zero, so the algorithm must converge.

3.2 Choosing a Classifier

A *classifier* simply maps an image into a class label. Many classifiers have been constructed, fine-tuned to the specific application at hand. Perhaps the first question to ask is whether there is one best classifier for a known distribution of classes. The best classifier is characterized by low probability of error. It turns out that the Bayes classifier is optimal; for background, see [2]. Although optimal, this classifier unfortunately can be very complex to implement and unwieldy especially for images of high dimension. As a result, the choice of classifier often boils down to less complex classifiers such as Linear Discriminant Analysis (LDA) or Classification and Regression Trees (CART). LDA is an example of a *parametric* classifier which assumes that the class distributions can be expressed in terms of common parameters like covariance matrices and expected vectors. CART is a *nonparametric* classifier which assumes nothing about the distributions at all. For our application, however, it is not so far fetched to assume a normal distribution for each class since the classes were constructed to make the distribution of each class as gaussian as possible. Although the covariance matrices for the classes are probably different which does weaken the effectiveness of LDA, the simplicity of LDA makes it a reasonable choice of classifier for our purpose.

As in *discriminant analysis* of statistics, LDA makes use of *scatter matrices* to represent the distribution of the images at hand. In particular, a *within-class scatter matrix* represents the scatter of samples around the mean of the class. Suppose we have a set of of N images, which has been partitioned into k classes. Let N_i denote the number of images in class i and M_i the mean vector of class i. The within class scatter matrix of our distributions then given by

$$S_w = \frac{1}{N} \sum_{i=1}^{k} N_i \Sigma_i,$$

where $\Sigma_i = \frac{1}{N_i} \sum_{j=1}^{N_i} (x_j - M_i)(x_j - M_i)^t$. The companion matrix, the *between-class scatter matrix*, is also used for LDA to represent the scatter of the means of all the classes around the mixture mean. Let the mixture mean be given by $M = \frac{1}{N} \sum_{i=1}^{k} N_i M_i$. This matrix is given by:

$$S_b = \frac{1}{N}\sum_{i=1}^{k} N_i(M_i - M)(M_i - M)^t.$$

For a given set of images, LDA tries to find a map $A : X \to Y$,where X is the image space and Y the feature space, that simultaneously minimizes the in-class scatter and maximizes the between-class scatter. We can think of the map that LDA furnishes as one that projects the classes onto a reduced space in which the classes are as distinct from each other as possible. To formulate a classifiability criterion, we need to translate these matrices into a number that reflects the relative degrees of scatter. There are several possibility, amongst which the most typical is

$$J = tr(S_w^{-1} S_b).$$

There are other criteria that are specifically fine-tuned for the cases of abnormal class distribution; these will not be considered here. What we need now is to find the map A that optimizes J in Y. To do this, we note that S_w is usually of full rank and hence has an inverse. We also note that S_b has rank $k-1$ since only $k-1$ of the M_i's are linearly independent, and hence the matrix $S_w^{-1} S_b$ does in fact exist with rank $k-1$. Associating features with eigenvalues, we know that $k-1$ eigenvalues of $S_w^{-1} S_b$ must be nonzero and all the others zero. We can view our feature space Y, then, as the space spanned by the $k-1$ eigenvectors corresponding to the nonzero eigenvalues found. The value of J in Y is simply the sum of these eigenvalues. This map does in fact preserve the criterion J since only the zero eigenvalues have been thrown out in the transformation and these do not contribute to the value of J.

To actually find A, we must solve the generalized eigenvalue problem,

$$S_b A = S_w A D,$$

where D is the diagonal matrix containing the eigenvalues to $S_w^{-1} S_b$. We then remove all the rows in A that correspond to the zero eigenvalues in D. Once this map is found, the classification of an image is simply a matter of finding the class whose transformed mean is closest to the transformed image.

LDA is the optimal classifier for distinct classes that are gaussian and have equal covariance matrices. The effectiveness of LDA is weakened when these criteria do not hold. For an ample number of images in the training set, LDA should still be effective in our case since k-bases assures a gaussian mixture model of distinct classes. For too few images, k-bases produces classes that are very likely too sparse for the distributions of the classes to exhibit "nice" behavior. In this case, LDA becomes extremely unreliable. The main drawback of LDA is that it manipulates the entire image. In the case of MRI images, the image space is of very large dimension which makes classification on the entire image computationally impossible since the diagonalization is an $O(N^3)$ operation. Additionally, directly manipulating images makes classification a very fallible process due to the presence of unwanted noise and distracting information contained within the images. A more glaring issue in light of our application is that applying LDA directly on the images requires the acquisition of the entire image, which is the ordeal that we wanted to circumvent in the first place. In order to make LDA even operational given images of high dimensions, it is imperative that the dimensionality of the images be reduced.

3.3 Local Discriminant Basis Algorithm

Our objective is to supply the classifier with a reduced set of features that contain the relevant information in the image. Since the relevant information hinges on class separability, we want to find the features that give us the most information about the distinct nature of the classes. The algorithm that we use to accomplish this is the Local Discriminant Basis algorithm (LDB) which finds a basis that most discriminates given classes. The process by which this algorithm selects the best basis from a library of bases is a modification of the Best Basis Algorithm of Coifman and Wickerhauser. The differing element is the information cost that is used. Rather than using a cost function that measures the efficiency of the representation, we want a function that measures discriminating power. For illustration purposes, let us consider two classes only. Let $\mathbf{p} = \{p_i\}_{i=1}^n$ and $\mathbf{q} = \{q_i\}_{i=1}^n$ denote the normalized energy distributions of images in the two classes. The sum of the elements in each of the sequences is equal to 1. The discriminant cost function should measure how differently $\mathbf{p}$ and $\mathbf{q}$ are distributed. A popular discriminant cost function is *relative entropy*, or *I-divergence*:

$$I(\mathbf{p},\mathbf{q}) \stackrel{\triangle}{=} \sum_{i=1}^{n} p_i log\frac{p_i}{q_i}.$$

This is easily modifiable to its symmetric counterpart, J, given by $J = I(\mathbf{p},\mathbf{q}) + I(\mathbf{q},\mathbf{p})$.

Let D be the discriminating cost function of choice. Extending this discussion to an arbitrary number of classes, the difference between the distributions of k classes, as represented by the sequences $\mathbf{p}^{(1)}, ..., \mathbf{p}^{(k)}$ can be measured by

$$D(\{\mathbf{p}^{(c)}\}_{c=1}^{k}) \stackrel{\triangle}{=} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} D(\mathbf{p}^{(i)}, \mathbf{p}^{(j)}).$$

The *local discriminant basis algorithm* is a fast algorithm that mirrors the Best Basis Algorithm for two-dimensional images using one of the discriminant cost functions detailed above. After the basis has been chosen, quantization can begin. To do this, we use *Fisher's Class Separability* index to measure the power of each basis function for the training set in order to make an informed decision concerning the coordinates to discard. For k classes, let N_i denote as usual the number of images in class i. Let $\Upsilon_c = ({x_1}^{(c)}, {x_2}^{(c)}, ...)$ be the images in class c where $\Upsilon_c(j)$ is the set of the coefficients of all the images associated to the basis function indexed by j. The power of basis function j for a given training set is given by:

$$F_j = \frac{\sum_{i=1}^{k} N_i \mid med(\Upsilon_i(j)) - med(\cup_{l=1}^{k} med(\Upsilon_l(j))) \mid}{\sum_{i=1}^{k} N_i mad(\Upsilon_i(j))},$$

where $med(\cdot)$ is the median function and $mad(\cdot)$ is the median absolute deviation function. In this manner, we can acquire a list of basis functions ordered by discriminating power, from which we can choose as many as is necessary for acceptable classification performance.

4 Summary

Figure 5 illustrates the algorithm developed in the discussion above. The algorithm consists of an

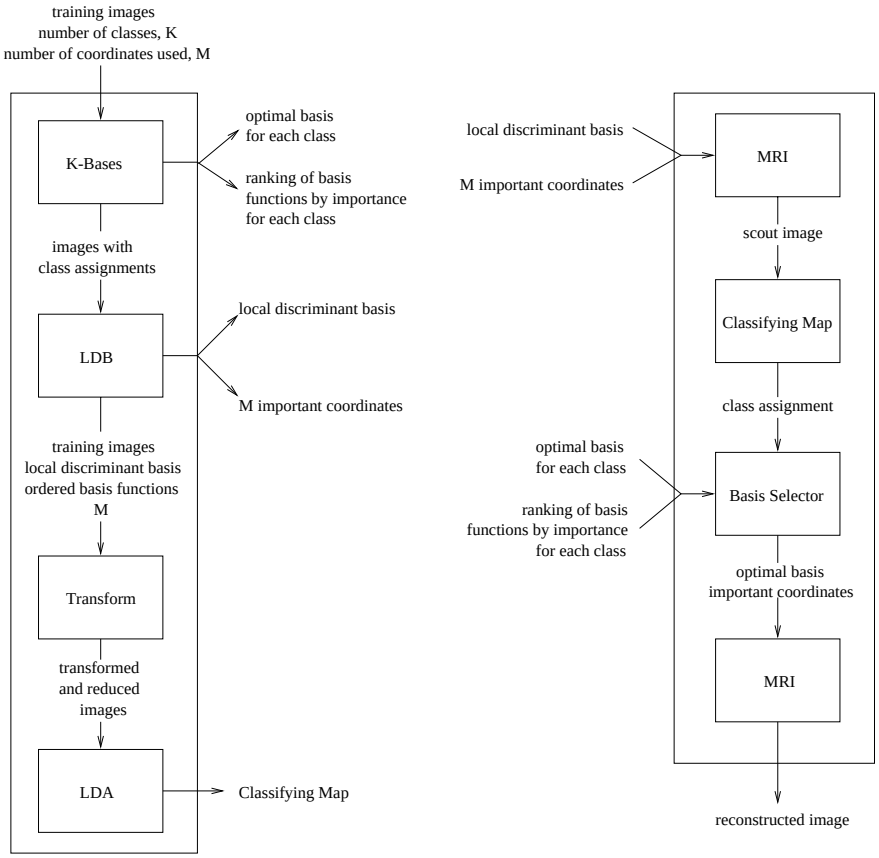


Figure 5: The off-line and on-line components of our algorithm for MRI images

off-line and an *on-line* process. The off-line process contains the "behind-the-scene" action; this provides all the machinery that is needed to acquire an image quickly. The on-line process is the actual process of acquiring a new image. The inputs to the off-line process are a training set of images that has already been acquired, the number of classes that we want, and a compression ratio that determines how many coefficients need to be discarded. Having specified these inputs, the training set is then sent through the k-basis algorithm which assigns each image in the set to a class. For each class, a basis is found by applying the best basis algorithm on the images in that class. This list of bases is kept to be used by the on-line process. The training set with the class assignments is next fed through LDB, which finds the best discriminant basis for the set. In LDB, we also compute Fisher's index for each of the new basis coordinates and order them according to discriminating power. The top M coordinates, which index the basis functions with the most discriminating capability, are kept for on-line purposes. After LDB, the training images together with the basis and the M important coordinates are sent through the transform. Each image in the training set is transformed into the new basis and then only the coefficients corresponding to the M most important coordinates are kept. These new and reduced images in the training set are finally sent through LDA, which constructs a classifier for the images of this localized area of interest.

With the off-line requirements complete, we can now take advantage of the simplicity and speed of the on-line mechanism to acquire an image in a similar region of the patient. We need acquire only the M important coordinates of the image in the discriminant basis to create a "scout image." This image, which can be acquired very quickly since M is much much smaller than the number of measurements needed to fill in the values of the 256×256 pixels, is then sent through the map furnished by LDA which in turn assigns a class to the image. Returning to the list of bases that we kept from the k-bases algorithm, we find the optimal basis for the particular class at hand. We then acquire the image using that optimal basis, omitting the coefficients with the least truncation error. The complete image can then be quickly reconstructed using the inverse wavelet packet transform.

The power of our algorithm is that the on-line task is fast and simple, while the computational complexity lies mostly in the off-line task that needs to be done only once for images in a certain region. Additionally, our algorithm has only made use of the flexibility of *existing* MRI hardware so no modifications in hardware design are needed.

To demonstrate the capabilities of our algorithm and to isolate the variables that have the most effect on its behavior, we tested the algorithm extensively on synthetic data. The synthetic data consist of multivariate gaussian distributions containing a specified number of classes with covariance matrices determined by different low-pass filters. Because we constructed the data, we can easily determine the variables that affect the behavior of our algorithm. The number of classes, the separation between the distributions, the size of the image and the number of images in the training set were the variables that were examined.

5 Results

We initially tested our algorithm on the synthetic images in the hopes of gaining a better understanding of the interaction between LDA and LDB. One problem we were immediately faced with was the limited number of MRI images that we could use as a training set. A primary objective while testing our algorithm was finding how much training data is enough for acceptable classification rates. Saito does not explore this question at all in his work, and uses instead a fixed training

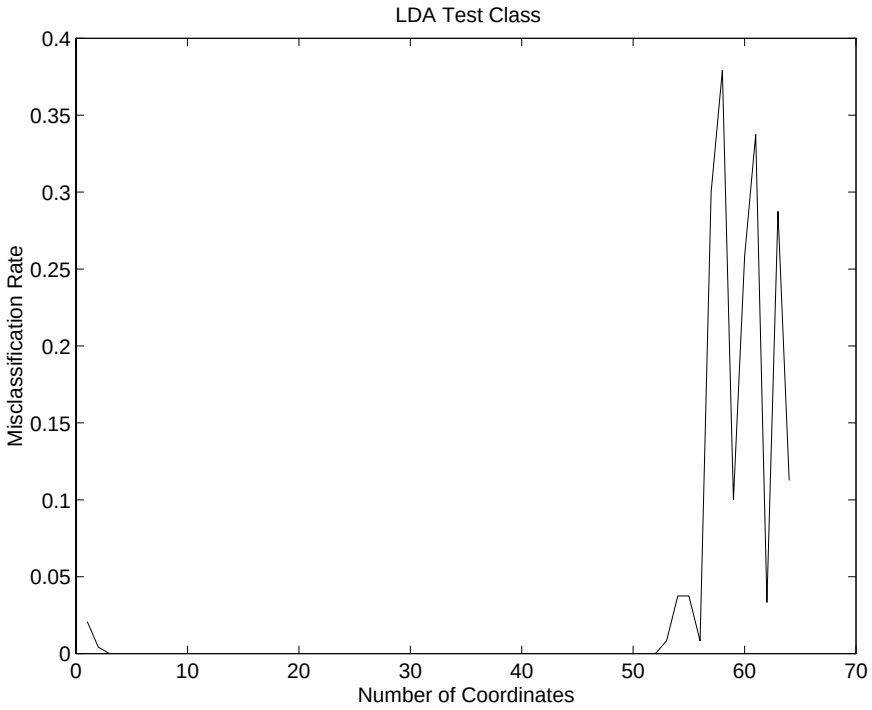


Figure 6: Unpredictable behavior in LDA as the number of coefficients exceeds the training set size (60)

set size that seems unnecessarily large.[3] By using the synthetic images and carefully controlling the parameters at hand, we found that as soon as the number of coordinates in the training images exceeded the number of unique training signals that we used, the behavior of LDA became extremely unpredictable. For well separated classes, the classification rates would go very suddenly from perfect to an unpredictable and seemingly random behavior, as demonstrated by Figure 6.

Other parameters were varied, such as the relative size of the testing and training sets, the image size, the separation between the classes, and the number of classes. None of these affected the rates so profoundly. Saito did not report evidence of this behavior in his paper; the obvious explanation for this is that the results he presented in his paper all used very generously-sized training sets which sidesteps the problem that we noted.

At this point, we wanted to know whether LDB could help push back this point of randomness in LDA. Upon extensive testing, we found that LDB had very little effect on the position of this instability point, and that the key parameter was the sheer number of training signals. Since LDB did not help in this respect, we wanted to determine the instances when LDB could improve classification rate. Our finding was that LDB helps when the classes are not well separated, and most importantly, when there is variation in the discriminating power of the coordinates.

The general trend to be expected as the number of coefficients increases is a steadily decreasing misclassification rate until the instability point is hit during which time the "random" behavior begins. Empirically, this instability point is the point at which the number of coefficients kept is equal to the number of images in the training set. In fact, the misclassification rates decrease

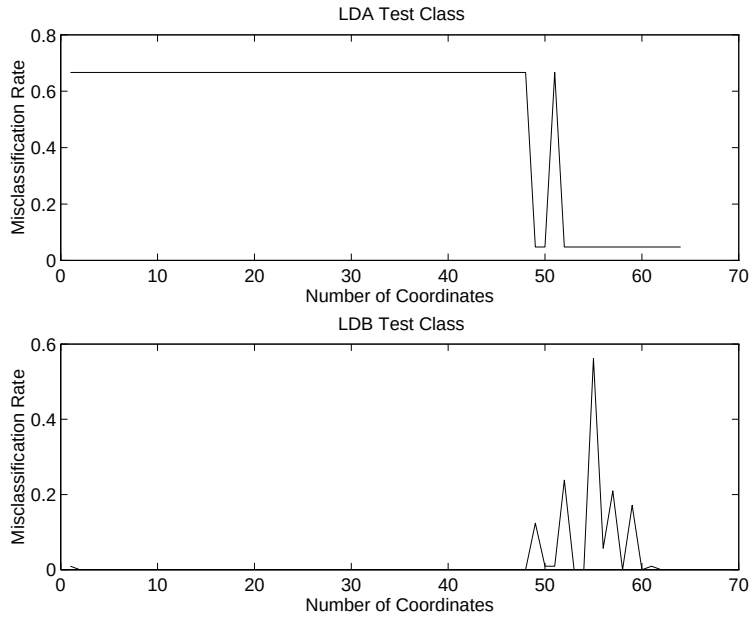


Figure 7: LDA vs LDB and LDA coefficient selection in action on image with borders, 45 training signals

steadily to a minimum, at which point it rebounds a little and hovers at a fairly steady level above the minimum. This behavior is not as contradictory as it might seem upon first glance. The first coordinates LDB picks are the most effective discriminant basis functions. As we pick more and more, the discriminatory power of the picked basis function drops. However, LDA has no way of weighting these basis functions according to this discriminatory power. It treats the coefficients equally. Thus, one might suspect that eventually the addition of more and more coefficients actually *dilutes* the discriminatory power of LDA.

We did not glean significant and consistent improvements over LDA in classification by using LDB on the synthetic images. We attribute this to the nature of the synthetic data. The basis that LDB found seldomly boasted significant levels of decomposition; that is, we ended up with the Dirac basis for the most part. Thus, any gains that LDB might have is directly dependent on the ability of the Fisher Median Separability Index to rate coefficients in discriminatory ability. The images, however, had fairly consistent information in all coefficients so LDB yielded very little improvement over LDA alone.

However, LDB was extremely effective when filler data was introduced into the synthetic images. In the extreme "screw" case in which the first couple hundred coefficients were zeroed out from each image, LDB was clearly advantageous. See figure 7. This contrived addition of zero content data is not a totally unreasonable since most images do in fact have regions (borders) that have the same color. Estimating the benefits of LDB and LDA for MRI images is difficult. It depends very much on how well the LDB basis is able to concentrate the discriminatory information into coordinate space. It is also not clear how an increase in training set size will affect the trueness of the LDB basis.

In order to gain insight into what LDB does to a group of images, we found the Local Dis-

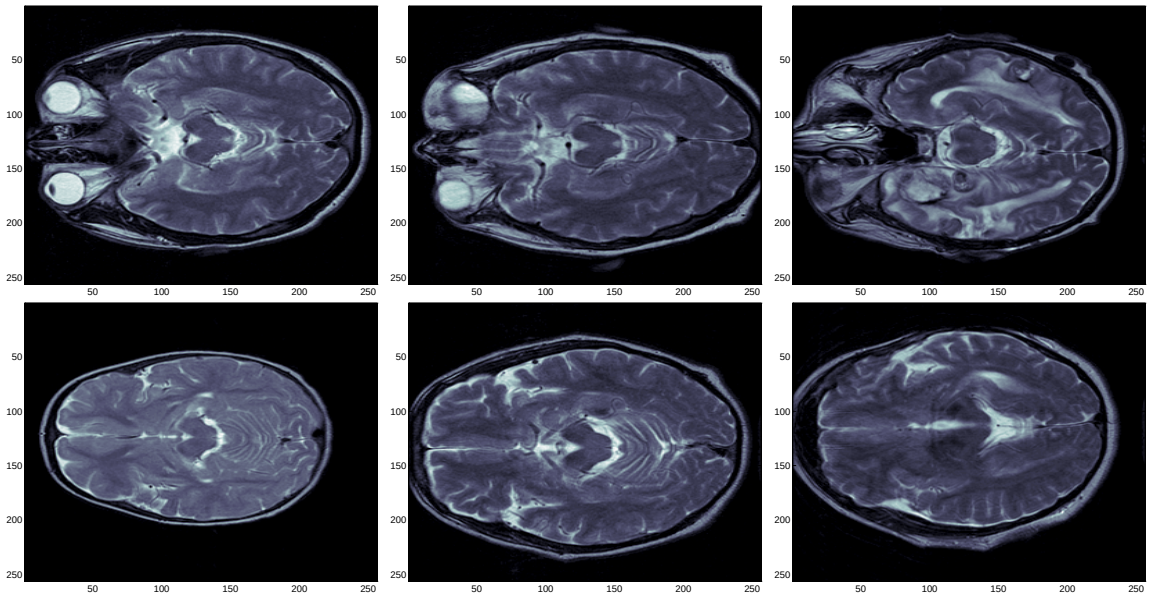


Figure 8: Sample training set of images with and without eyes

criminant Basis for subsets of size 10, 18 and 36, of our 36-image training set. We then ran the inverse wavelet packet transform repeatedly on a test image, using the LDB for each given set and increasing quantities of significant coefficients. The resulting images–located in Appendix B– provide us with an idea of what features LDB selected as important, and thus how well LDB was trained. See figure 8 for a subset of the training set images. As can be seen, with 10 training images, between 5000 and 20000 coordinates (out of 65536) are needed before the basis functions reveal a feature that the human eye might recognize as a feature. In this case, the eyes seem to be the primary focus. On the other hand, the 18 image training set has regions that can clearly be identified as eyes starting from between 100 and 500 coordinates. It appears, then, that the size of the training has an enormous effect on the validity of the LDB basis. In fact, we might even make the conjecture that even with LDB trained on 18 images, the top 500 coordinates might be enough to classify quite well. That is, forgetting that LDA itself will require at least 500 training images to accompany those coordinates if we want to avoid instability. With 500 training images, however, LDB is likely to be even more effective.

The question then remains: how feasible are the calculations on a large training set? The answer is that they are quite feasible, just not quick. At present, creating a packet table for an image takes about six minutes on a 133 MHz Dec Alpha. With 500 or so images, it would take two or three days to complete. However, the packet table code could very easily be rewritten in C, for an estimated running time of roughly twenty seconds. Furthermore, multiple computers could be used to do this operation, so the time could be reduced to a fairly negligible amount. The other time consuming operations are the k-bases algorithm and the algorithm to implement LDA. The k-bases algorithm could be accelerated best by choosing good starting means. The algorithm for LDA is bound by the simultaneous diagonalization process, which executes in $O(N^3)$. It takes approximately five seconds for sixty coefficients. For ten times as many coefficients, we can estimate that it would

take about 1000 times longer. 5000 seconds is also a fairly reasonable quantity of time. Since the off-line algorithm is a one time cost, it does not seem to be terribly restricting. However, it would probably be useful to have accelerated versions of the algorithm for further exploration.

5.1 Future Work

The results should be extended for actual MRI images, using a large training set. LDA seems to be problematic because of its running time, and its general instability. The main culprit regarding the slow running time of LDA is the packaged function that we used for simultaneous diagonalization. Further exploration concerning this function as well as the approximate versions might be useful. It might also be interesting to retain LDB but to use another classifier. As mentioned before, a nonparametric classifier such as CART has no assumptions about the nature of the class distributions which might be useful since the covariance matrices of the classes are very likely to vary. Another approach that bypasses the need for a classifier altogether is to convert the k-bases means into the new basis representation and then to classify the transformed signals according to closest mean. That is, we can classify by finding the distance from the transformed means in the same way that the k-bases algorithm originally classifies the signal. This would presumably be very near optimal, since it is after all k-bases' classification that we want to mimic in order to get the optimal basis. To make life easier for *any* classifier as well as for the clustering in the k-bases algorithm, it might be worthwhile to investigate the possibility of positioning and scaling the images in the training set so that they are self-consistent. For example, the features for class separability should not be the margin size of the frame around the actual image. Finally, we could use the symmetric J-divergence instead of the asymmetric I-divergence for the LDB selection algorithm. Better local discriminant bases might be found in this way.

A Code Placement

The code is loosely organized into several directories, according to the algorithms each function is related to. Most of the code has been written in MATLAB. The principle bottleneck functions have been rewritten in C.

/

`lda/`	Linear Discriminant Analysis source
`ldb/`	2 dimensional Local Discriminant Bases source
`pita/`	K Bases source
`testing/`	LDA and LDB glue source
`WaveAlt/`	WaveLab and Matlab Replacement Functions (MEX treasure pile)
`kitsch/`	Tex and EPS source for this document
`results/`	Results of testing runs

/lda

`ldaMap.m`	Offline part of LDA. Given a training set (each signal, prefixed by a class), produces the linear discriminant mapping matrix. The means of the classes are also produced. Given a new signal, one can classify it by multiplying it with this matrix, and finding the class whose mean is closest.
`ldaProper.m`	Online part of LDA. Given the map, the means (both generated in ldaMap), and the signal to classify, returns the class.
`classCovarTwo.m`	finds the "between" and "within" class scatter matrices required for ldaMap
`sampleCovariance.m`	used to created "within class scatter matrices" for classCovarTwo

/ldb

`ldbTwo.m`	Given, the signals and their class assignments, determines the best discriminant basis. Calls MultiClassDis and Best2dBasis (from Wavelet library), which basically do a modified BestBasis on the packet table sum generated by TFEMTwo.
`ldbProperTwo.m`	Takes class members and some optional parameters (which allow caching of values), returns the local discriminant basis, the transformed signals in the new basis, the array of the discrimination power of each basis function, a list of the most discriminating coordinates in decreasing order, and the original signals, transformed into the new basis, and rearranged with most significant coordinates first. In other words, this hydra like function is just about as heinous as realloc or fcntl.

`TFEMTwo.m` — Calculates the spatial-frequency energy map for the class; eg the sum of all the packet tables for a class of signals. If called with three parameters, it calls Calc2dPktTable from the Wavelet library. Otherwise, it loads the packet table off disk, which usually takes less time. The variable name of the saved packet table in the file is 't'.

`fastFisherMed.m` — Given all signals and their class assignments, calculates the discriminant power of each individual basis function. Outputs a vector that is the size of the signals of ClassMembers where the ith entry in the vector corresponds the the discriminating power of the ith coordinate. Used by ldbProperTwo.

`MultiClassDis` — Driver stub which calls megaCalc2dTree, which is a modified version of Calc2dStatTree which implements the LDB cost function instead of the usual Best Basis cost function.

/pita

`KBases Code` — Slightly modified from the version by Sumit Chawla. Uses FastFPT2_WP (and thus cached packet tables) to quickly obtaining the representation of a signals in a basis.

/testing

`runldbP2.m` — Script that handles most of the processing LDB is involved with (ie mostly processing that would be done off-line in practice). Creates x classes of synthetic images, each with y signals, selecting z percent of each class as a training set. The distances between the means of the images are set in generatedata. Packet tables for the training set are create en masse. This is the major bottleneck at current time. If Calc2dPktTable (from the Wavelet library) could be rewritten, this whole LDA/LDB algorithm would run quite much more quickly. The calculation of the packet tables for the training set is a parallelizable operation. k_bases is run to determine the training set classes. convertAssignments is called to arrange the signals sequentially by class and then to set up links to the cached packettable files according to the new positions of the files. ldbTwo and ldbProperTwo are then called. Finally all of the signals designated as test signals are FPTd (another time intensive operation), for preparation for runldaP2.

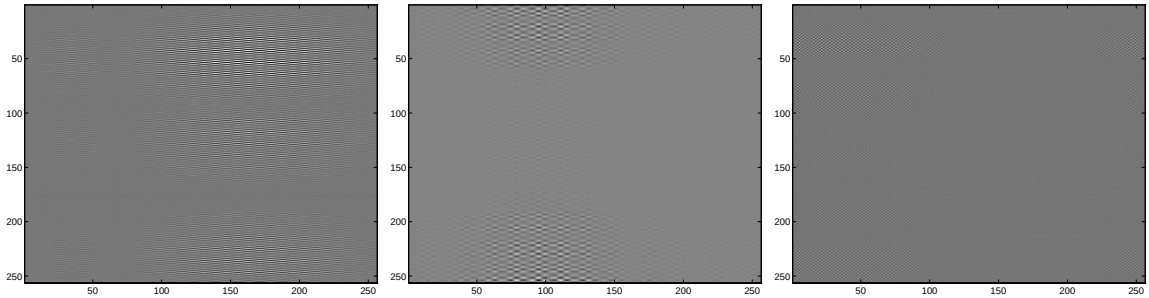
runldaP2.m	For a given range of number of coefficients, calculates the training and testing set misclassification rates for LDA on the first k coefficients of the image, and for LDB on the first k most significant basis functions. These rates are then graphed. Note that the most dangerous function (time wise) is called here – ldaMap, which calls the eig function (which runs in $O(N^3)$ where N is the number of coefficients being used.)
ldaldbProper.m	Takes the LDB basis, a list of the most important basis function indexs, the LDA "map" matrix, the classMeans, and the signal to classify. transforms it into the new basis, extracts the most important coefficients, and feeds them into the LDA algorithm for classification.
convertAssignments.m	Given a list of assignments and signals, arrange the signals sequentially by class and then to set up links to the cached packet table files according to the new positions of the files.
sic.m	Small test signal set, 2 classes.
sic2.m	Another small test signal set, 2 classes.
smallimageclass.m	Another small test signal set, 2 classes.
transformSignalsTwo.m	Transforms a group of signals into a basis. Provides optional parameters so that cached packet tables can be used instead of calling WaveLab's FPT2_WP.
CreateTrainingSet.m	Makes links to all of the files in "sourceDir." A certain percentages of these links will be located in "trainDir," the rest in "testDir."
generatedata.m	Creates X clusters of size Y synthetic 2d images.
imgldbP2.m	Version of runldaP2 which uses precreated images (for instance, mri data) instead of generating synthetic images.
WPDirectory.m	Creates packettables for all of the files in a given directory.
eyesNoeyes.m	Tool which cycles through images in a directory and saves them in a new directory according the user specified class. The files are saved such the image data is stored in the variable t, instead of the filename. (This is important because the names of the files change.) This is also a shunt between the data format of the MRI images and our own internal file representation.
createLdbFI.m	Script that creates postscript images of a signal in the LDB basis using the most important x coordinates where x is a variety of values. Uses ldbVisualize. Used to see what features LDB is picking.
pitcw.m	Pretty picture viewer. Takes two dimensional image. Performs automatic scaling of values for palette.
createSynthFiles.m	Calls generateData to create synthetic images. Saves them with a filename that indicates the class they belong to.
ldbVisualize.m	Takes the rearranged transformed signal, and de-arranges it according to the mapping that the list of importantCoordinates provides. Performs the inverse wavelet packet transform and displays the image using pitcw.

/WaveAlt

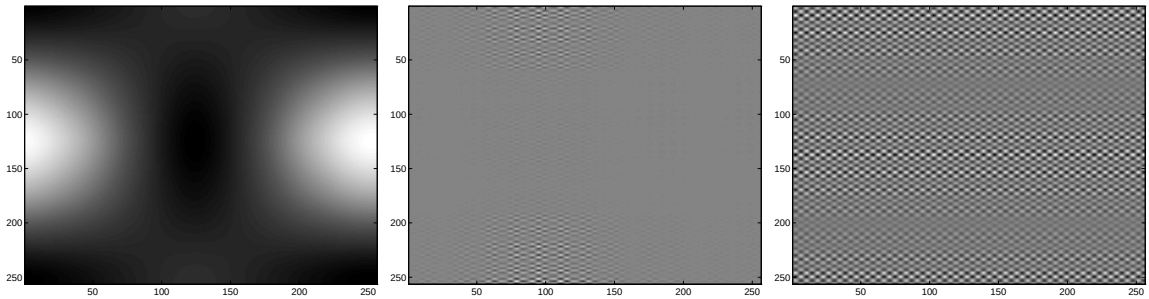
`FastFPT2_WP.c`	Fast version of FPT2_WP. Uses a Packet Table to compute the representation of a signal in a given basis. Very fast in comparison to the function it replaces.
`megaCalc2dTree.c`	Fast version of Calc2dStatTree, but for the LDB cost function only.
`refinedMed.c`	Fast version of MATLAB median function. Uses O(N) algorithm instead of O(NlgN) algorithm. Rocks.
`FPT2_WPN.m`	Internal version of FPT2_WP.m – now identical.
`Best2dBasis.mex`	Automatically generated MEX version of WaveLab function.
`Unpack2dBasisCoeff.mex`	Automatically generated MEX version of the WaveLab function. Still really horribly slow. Used to verify the correctness of FastFPT2_WP.

B Images Reconstructed with LDB Features

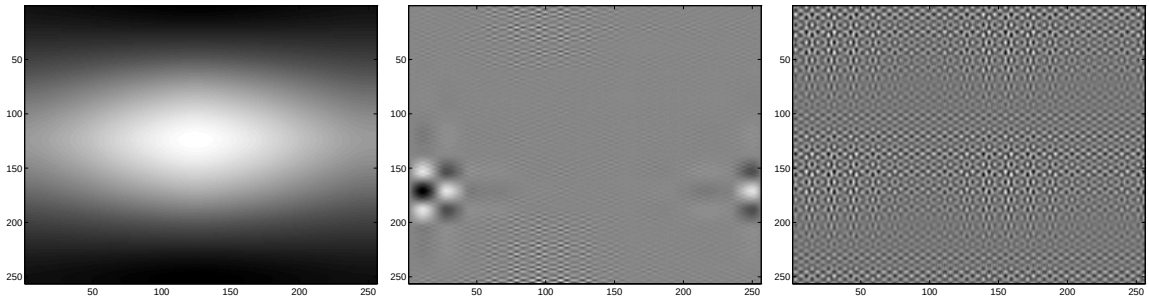
Here are the reconstructed images using successively more features. Three different bases are used to qualitatively ascertain the effects of larger training sets on LDB. The training sets used were of size 34, 18, and 10. Reconstructed images which highlight the detail of the brain image (in particular the eyes) most closely with the fewest coordinates are presumably better. The two classes used were "images with eyes" and "images with no eyes."



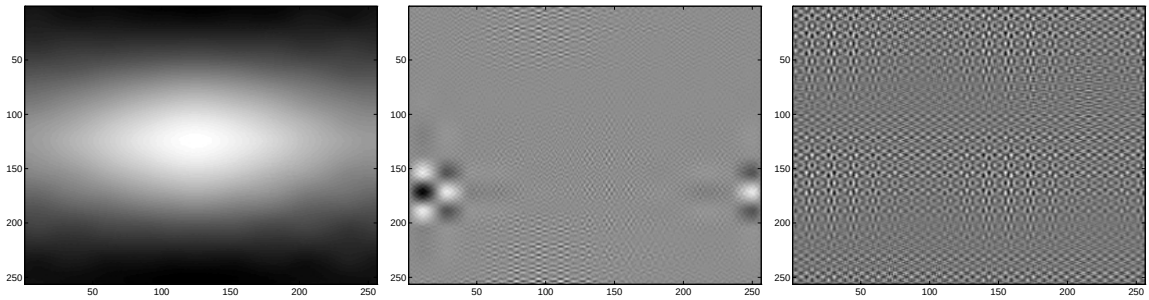
Reconstruction using most significant coordinate



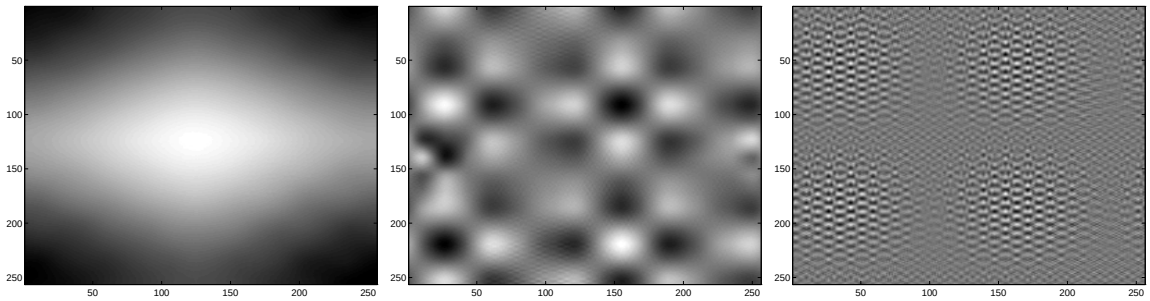
Reconstruction using most significant 2 coordinates



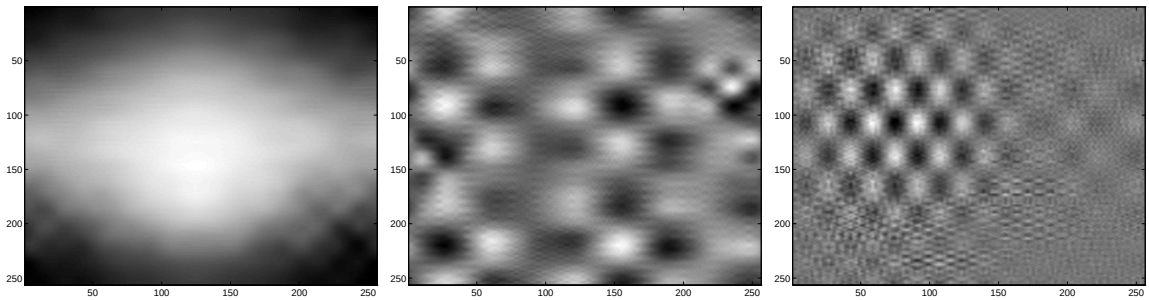
Reconstruction using most significant 5 coordinates



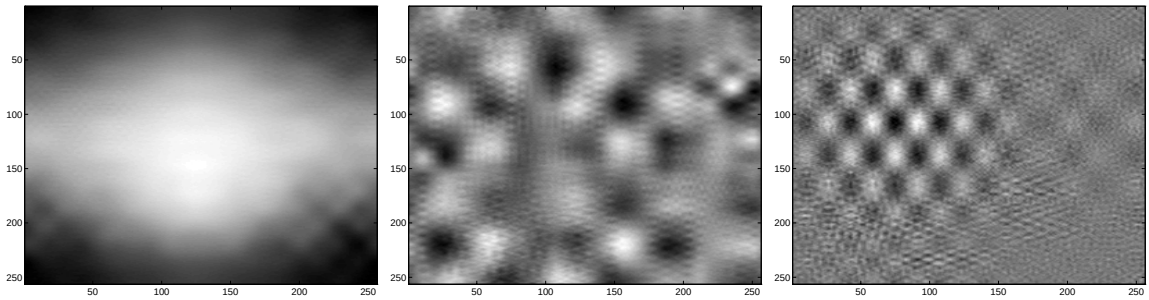
Reconstruction using most significant 10 coordinates



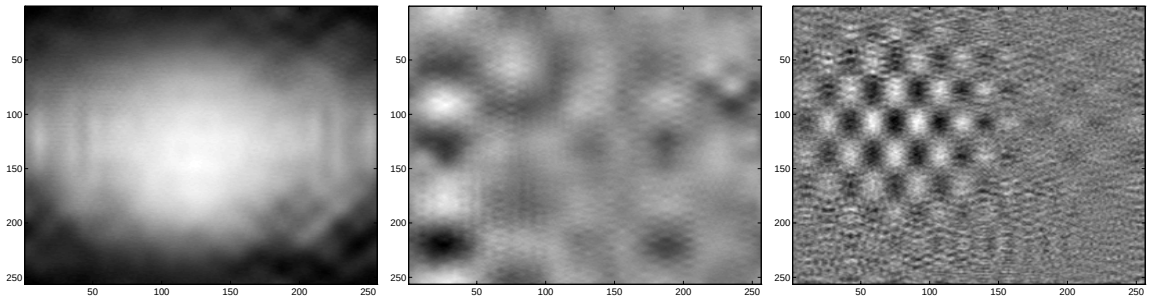
Reconstruction using most significant 20 coordinates



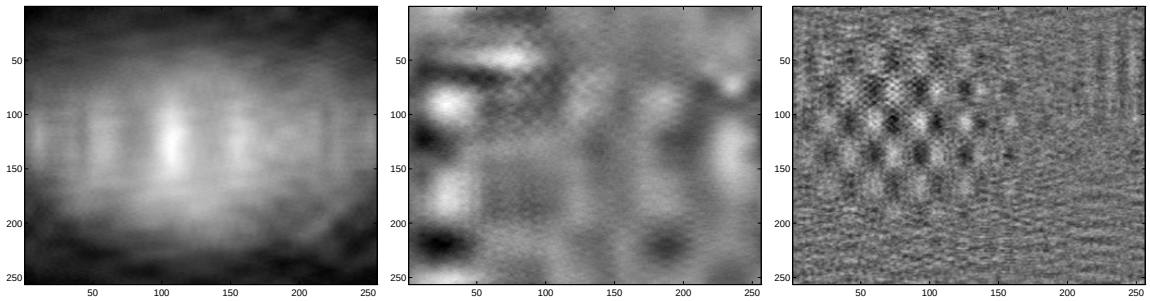
Reconstruction using most significant 50 coordinates



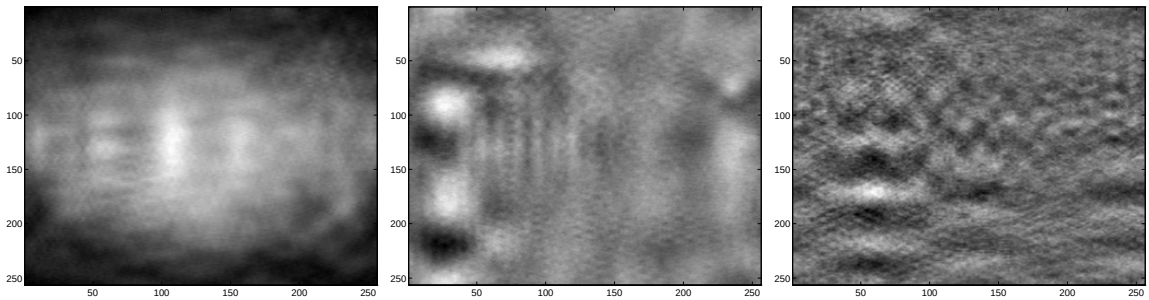
Reconstruction using most significant 100 coordinates



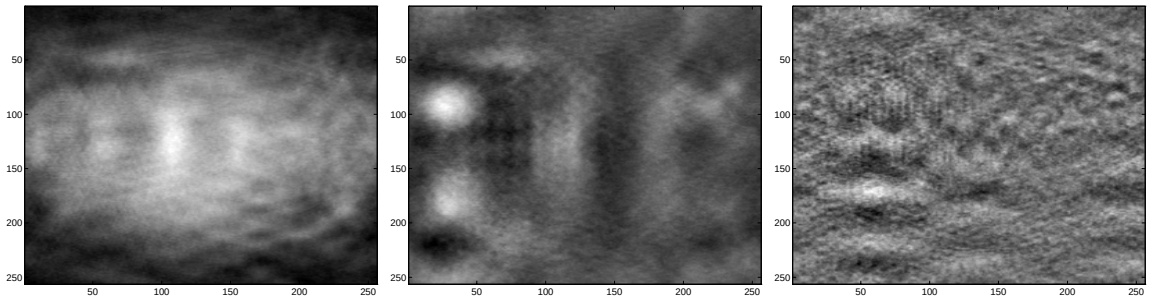
Reconstruction using most significant 200 coordinates



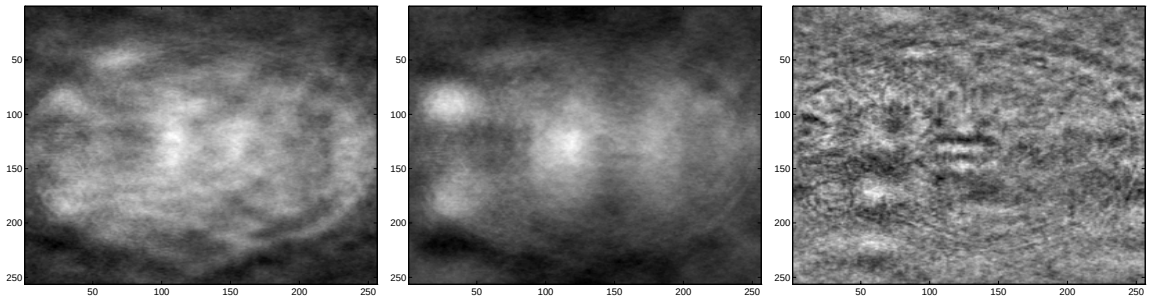
Reconstruction using most significant 500 coordinates



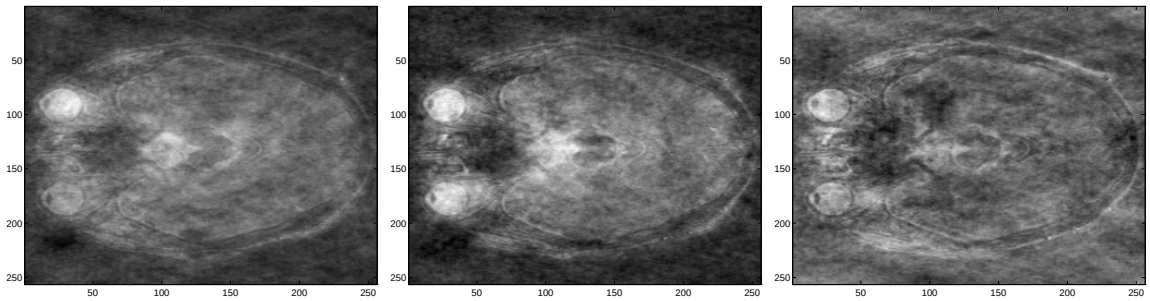
Reconstruction using most significant 1000 coordinates



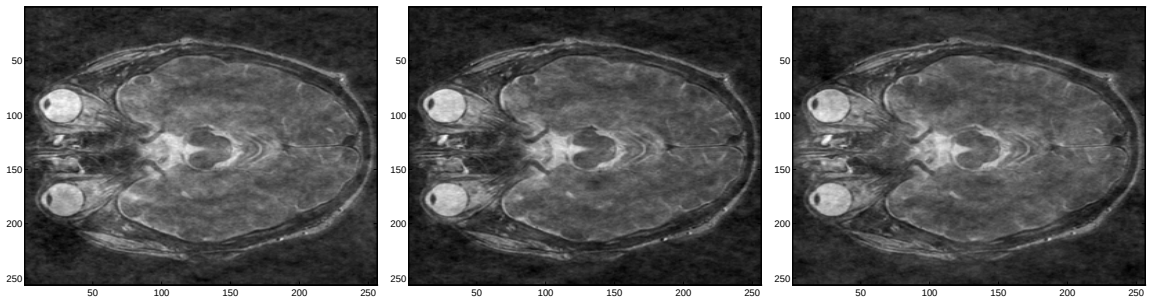
Reconstruction using most significant 2000 coordinates



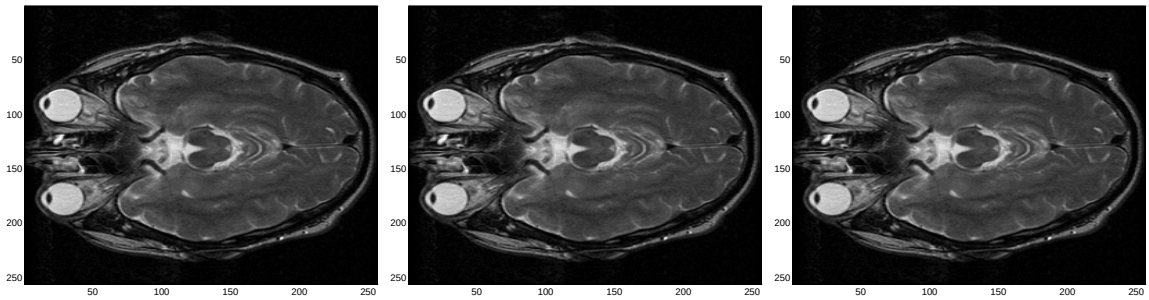
Reconstruction using most significant 5000 coordinates



Reconstruction using most significant 20000 coordinates

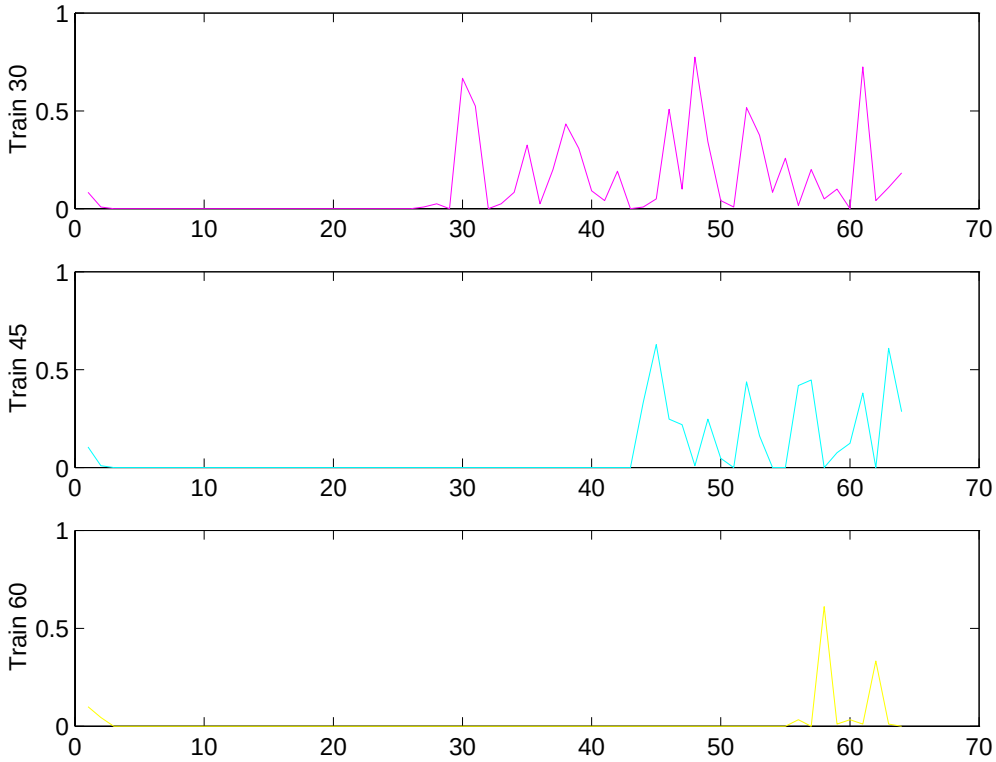


Reconstruction using most significant 50000 coordinates

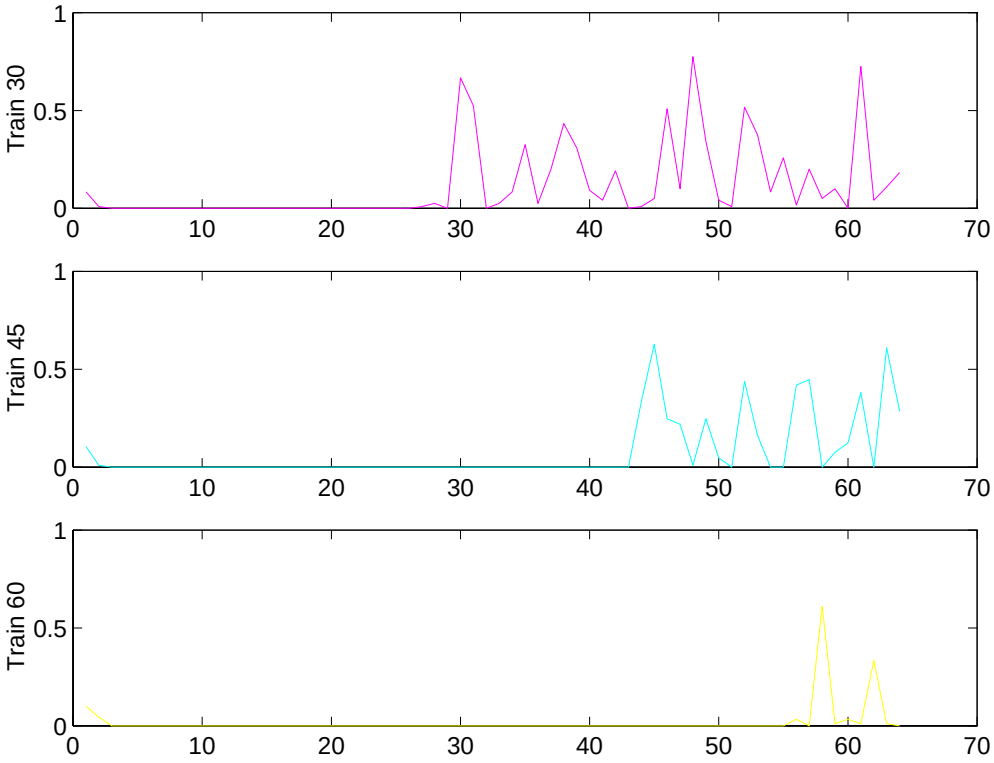


Reconstruction using all (65536) coordinates

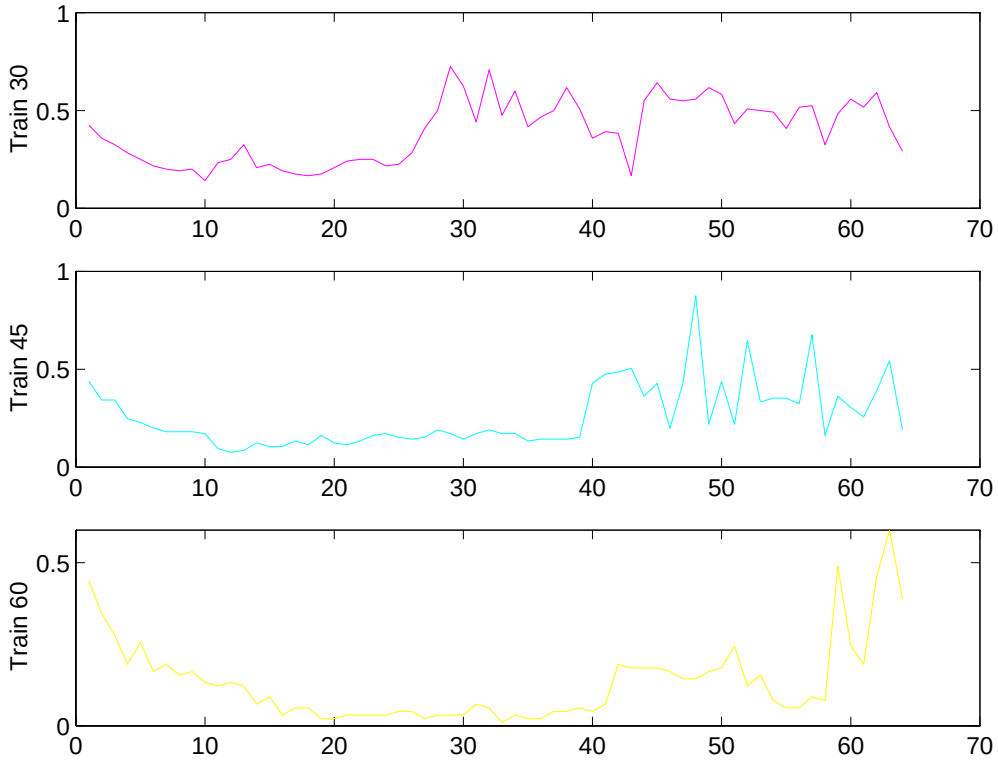
C LDA and LDB/LDA Misclassification Rates



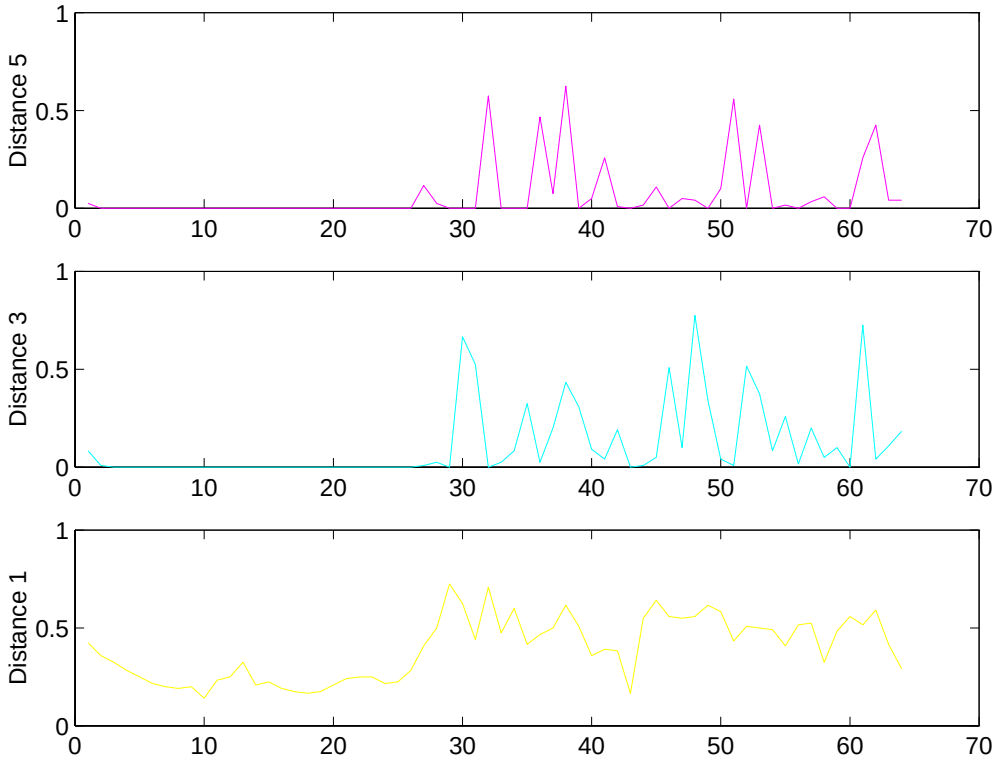
LDB misclassification rates with respect to training set of various sizes, 3 classes, $32x32$ image, 150 test signals, class separation of 3. Note the apparent randomness that occurs when the number of coefficients reaches the number of training signals.



LDA misclassification rates with respect to training set of various sizes, 3 classes, $32x32$ image, 150 test signals, class separation of 3. Note the apparent randomness that occurs when the number of coefficients reaches the number of training signals.



LDB misclassification rates with respect to training set of various sizes, 3 classes, $32x32$ image, 150 test signals, class separation of 1. Note the apparent randomness that occurs when the number of coefficients reaches the number of training signals.



LDB misclassification rates with respect to class separation, 3 classes, $32x32$ image, 150 test signals, 30 training signals.

References

[1] Healy, D.M. Warner, D.W., Weaver, J.B. *Adapted Wavelet Encoding in Functional Magnetic Resonance Imaging* Dartmouth College.

[2] Fukunaga, K. *Statistical Pattern Recognition. Second Edition* Academic, Boston. 1990.

[3] Saito, N. *Local Feature Extraction and Its Applications Using a Library of Bases* Yale University. 1994.

[4] Wickerhauser, M. *Picture Compression By Best-Basis Sub-Band Coding* Yale University. 1992.