1-1-2013

A Study Of Three Dicitonary Learning Algorithms

Zhiliang Xing
Ryerson University

A STUDY OF THREE DICTIONARY LEARNING ALGORITHMS

by

ZHILIANG XING, B.Eng.,
Shanghai Normal University, Shanghai, China, 2010,
University of Dayton, Dayton, USA, 2010,

A project
presented to Ryerson University

in partial fulfillment of the
requirements for the degree of
Master of Engineering
in the Program of
Electrical and Computer Engineering.

Toronto, Ontario, Canada, 2013

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Title: A STUDY OF THREE DICTIONARY LEARNING ALGORITHMS

Student Name: ZHILIANG XING, B.Eng.,
Shanghai Normal University, Shanghai, China, 2010,
University of Dayton, Dayton, USA, 2010

Program: Electrical and Computer Engineering

Name of University: Ryerson University

Sparse Representation is a topic that has been gaining popularity in recent years due to its efficiency, performance and its applications in communication and data extraction fields. A number of algorithms exist that can be used to implement sparse coding techniques in different fields which include K-SVD, ODL, OMP etc. In this project one of the most popular sparse algorithms, the OMP (Orthogonal Matching Pursuit) technique, is investigated in depth. Since OMP is not capable of finding the global optimum, a Top-Down Search (TDS) algorithm is proposed in this project to achieve much better results by sacrificing the execution time. Another contribution of this project is to investigate the properties of dictionary by modifying the frequency and shifting the phase of a standard Discrete Cosine Transfer (DCT) dictionary. The results of this project show

that the performance of sparse coding algorithm still has room for improvement using new techniques.

Keywords: Sparse coding, OMP, Top-Down search algorithm

Acknowledgments

I wish to express my sincere gratitude to Dr. Kaamran Raahemifar for his knowledgeable guidance and patience. Thank you for working closely with me on my MEng project. I appreciate all the encouragement you give and all these regular meeting time. This MEng project would not be finished on time without your kindly help.

Finally, I would like to appreciate my family and my best friend Sheldon Mark Foulds for their nonstop and warm support.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Motivation

Sparse approximation is representation that accounts for all information of a signal in terms of a linear combination of small number of elementary signals called dictionary atoms [1]. These atoms are often selected from a dictionary that is fixed in nature. Dictionary learning is a process to find this sparse approximation for certain signals. In the past few years dictionary learning techniques have been gaining popularity due to their efficiency and high performance making them suitable for applications such as noise reduction, compression, feature extraction, pattern classification and blind source separation [2, 3].

This technique for finding a sparse representation from a small number of significant coefficients is referred to as Sparse Coding. While the decoding process remains relatively straightforward the real challenge lies with the encoding process where the input vectors have to be represented with the least possible number of atoms. This proves to be an NP-hard problem because the number of combinations from the available solution space is usually extremely large. Thus, considerable effort has been made to find near-optimal schemes that allow us to acquire a solution that while may not be the global optimum in the solution but at least a near optimum value. Some examples of these sub-optimal schemes

include Matching Pursuit, OMP, K-SVD, ODL, and online dictionary learning [4,5,6,7]. Another popular branch of dictionary learning tried to learning an image-based dictionary based on the picture itself [8,9,10]. The motivation of this paper is centered on the limitations of the OMP algorithm and what can be done to improve efficiency and accuracy of sparse coding.

The contribution of this paper is as follows: (1) An in-depth evaluation of the popular OMP algorithm and a study of its performance compared with that of a technique called the Top-Down search algorithm. The top-down search algorithm is a sparse coding technique that uses the curve fitting approach to find optimal solutions while sacrificing the execution time. The Top-Down search algorithm will allow us to independently verify if the OMPs abilities to accurately represent input images is severely limited or not and whether the OMP is actually able to find values that are closer to the global optimum or not. (2) The second contribution of this paper is aimed at understanding the effects of modifying the dictionaries which the algorithm uses to generate its results. Instead of using a fixed and known dictionary, the modification of dictionary tries to improve the quality of dictionary. Three different approaches are implemented in this paper; two approaches aim to expand the standard DCT dictionary using two different strategies, while the last approach apply a popular heuristic algorithm called Particle Swarm Optimization Algorithm to “learn” a dictionary iteratively.

1.2 Sparse Representation

Sparse representation is the “core” of dictionary learning algorithms. The idea is to represent a large amount of data using a linear combination of small amount of basis, therefore the data can be transmitted, saved, or processed in an efficient way. Many practical and useful applications can be implemented based on the idea of sparse representation. For instance, high quality images are usually large files in modern computer systems. If an image is sent from one device to another device through wireless network, instead of sending the original image with all the data pixel by pixel, through sparse coding only a small amount of information is encoded on the transmitter side and decoded on the receiver side. Figure 1.1 shows an example in a real implementation. The original test picture is sparse represented using a linear combination. The picture is firstly divided into small 8x8 blocks, and each block is represented by a linear combination of 8 basis. The major issue of the sparse representation is how to find the global optimum or a sub-optimum solution which is really close to the global optimum.

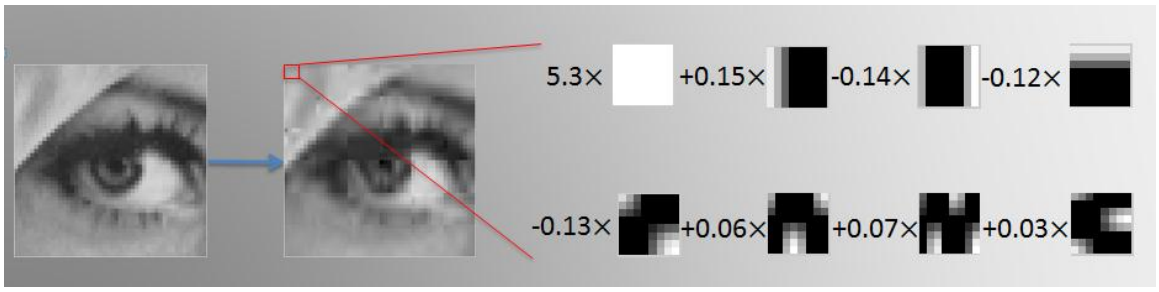


Figure 1.1: Sparse representation example

1.3 Thesis Organization

This thesis is organized in a structured approach to make it easier to understand the concept of sparse representations. In Chapter 2 a short survey of the OMP algorithm and the PI-DCT dictionary used in the tests are presented. In Chapter 3 background concepts necessary to understand the study are presented; this includes the concepts of the Top-Down dictionary and the dictionary properties under investigation. In Chapter 4 the simulation results of the tests conducted are presented along with a flow of the simulations to describe how each test progresses. In Chapter 5 the results of the simulations are presented, along with a detailed analysis and comparison between the different tests. Finally this thesis is concluded by presenting future work and proposals for increasing the performance of the sparse coding techniques.

Chapter 2

Survey

The sparse coding process begins with an input image which is taken and divided into windows each of 8X8 pixels. It is represented as a set of input vectors $\vec{x} \in \mathbb{R}^d$. This window is, then, mapped onto a pre-compiled DCT dictionary in terms of dictionary atoms given by $\vec{b}_1, \ldots, \vec{b}_n \in \mathbb{R}^d$. This mapping is a linear combination of dictionary atoms which represent discrete cosine signals with different frequencies. Using these atoms in the dictionary the sparse coding algorithm, then, calculates coefficients or weights $\vec{s} \in \mathbb{R}^n$ for representing the input image window in terms of the dictionary atoms. Thus, the overall input image is represented as $\vec{x} \approx \sum_j \vec{b}_j s_j$ or $X = B \cdot S$ in matrix form, once a window is represented within the sparsity restrictions and the error constraints; this process is repeated until the whole picture is represented sparsely. Using these representations it is then able to reconstruct the original image with the sparse terms, the quality of this reconstructed image is judged by subtracting the sum of the sparse terms from the original vector to receive a reconstruction error $\vec{x} - \sum_j \vec{b}_j s_j$. The objective throughout this study is to reduce this reconstruction error as much as possible. To limit the scope of this thesis only a brief description of the sparse signal representation of images is provided since the topic is extensive, the basics of image processing techniques and the further reading on this can be found in [11, 12].

Before going further it is important to clearly define the concept of sparsity factor. The sparsity factor represents the number of non-zero terms that can be used in the linear equation to represent the sample window [13]. For example, if an image is transmitted from the sender to the receiver across a network, it is impractical to transmit all the data in the input image. A more practical approach would be to transmit data that is sufficient enough to reconstruct the image at the receiver. The size of the data that can be used to represent the image, then, becomes really important because a smaller size linear equation leads to much faster processing between transmitter and receiver. This leads to a dramatic decrease in processing time.

2.1 Orthogonal Matching Pursuit (OMP) Algorithm

OMP algorithm happens to be one of the most widely used sparse coding algorithms because of its efficiency and high accuracy [14,15,16]. It is a matching pursuit based greedy algorithm which is centered on three important features; residue, generalized dictionary atoms, and weight. In [17,18,19], OMP algorithm is optimized to be more efficient. However in this project the basic OMP is implemented and tested. Given the input vector x, the OMP algorithm finds us X which is a reconstruction of the original input signal in terms of a weight matrix and the dictionary matrix. The relation between the weight matrix S and selected dictionary atoms B, is given by the following formula:

$$X = B \cdot S \quad where\ S\ is\ the\ weight\ matrix \tag{1}$$

where *S* is the weight matrix.

The OMP first takes one column x from the input matrix X as the input to itself; this vector is also chosen as the initial residue. The residue is defined as the difference between the reconstructed vector and the initial input vector. The initial residue is defined as

$$R_1 = x, \tag{2}$$

The algorithm, then, must choose an atom B_i from the given dictionary B which is a best match to the original vector. This is done by calculating and finding the maximum inner product of the generalized dictionary atoms and the $residual$:

$$\max\{B_i \cdot residual\} \tag{3}$$

Once the product for all dictionary values is calculated a matrix with the best values is chosen using the following formula:

$$s = (D^T D)^{-1} D^T x, \tag{4}$$

where *s* is a weight vector for all selected dictionary atoms, *D* is the matrix of all selected dictionary atoms from the first iteration till the current one, and *x* is the current input vector.

This formula makes sure that the residual is always orthogonal to the current selected dictionary atom in the current iteration; this is one of the most crucial differences between OMP and matching pursuit. Next, it subtracts the current reconstructed vector from the original vector to update the residue.

$$R_{n+1} = R_n - D \cdot s \tag{5}$$

The algorithm continues to do the above process until the constraints are met. This makes the accuracy of the OMP algorithm high, but the main reason for choosing the

algorithm is its efficiency of getting results in shorter period of time. The algorithms main steps are outlined in Figure 2.1.

Orthogonal Matching Pursuit (OMP) algorithm:

1 $s = OMP(B, x, sparsity)$
2 $w = 0, residual = x, sparsity = 0$
3 *while not finished*
4 $\quad inner\ product = B^T \cdot residual$
5 $\quad find\ k: |inner\ product(k)| = max_j |inner\ product(j)|$
6 $\quad s(k) = (D^T D)^{-1} D^T x$
7 $\quad residual = residual - D \cdot s(k)$
8 $\quad sparsity = sparsity + 1$
9 $\quad if\ ||residual|| \leq\ constraint\ then\ finish$
10 $\quad if\ sparsity = constraint\ then\ finish$
11 *end*
12 *return*

Figure 2.1: OMP algorithm

To illustrate the reason why OMP has the ability to reach higher accuracy than the well-known Matching Pursuit (MP) algorithm, the major difference between MP and OMP is explained in a 2D space example in figure 2.2. There three vectors, $\alpha \times d_{\gamma n}$, R_n, and R_{n+1}. $\alpha \times d_{\gamma n}$ is a dictionary atoms with its weight. R_n is the residual of the current iteration, while R_{n+1} is the residual of the next iteration. The ideal of MP algorithm and OMP algorithm is gradually adding dictionary atoms with weight in each iteration to reduce the residual. In figure 2.2, it is obvious that R_{n+1} is shorter than R_n, and will be further reduced in the next iteration. The only difference between OMP and MP is when MP update the weight, it directly uses the inner product value $\langle R_n, d_{\gamma n} \rangle$, while OMP uses the equation (4) to make sure that $\alpha \times d_{\gamma n}$ is orthogonal to R_{n+1}. The advantage is that in each iteration, OMP always find the shortest distance as shown in figure 2.2 (a).

(a) (b)

Figure 2.2: (a) Matching Pursuit Algorithm (b) Orthogonal Matching Pursuit Algorithm

2.2 PI-Dictionary Property

The original PI-DCT dictionary is a 2D discrete value dictionary that provides values as a basis for image reconstruction [20]. The dictionary is composed of two cosine functions with its frequency ranging for 0Hz (DC) to approximately 4Hz. The PI-DCT dictionary uniformly distributes the frequency with the default value of 1/16 Hz.

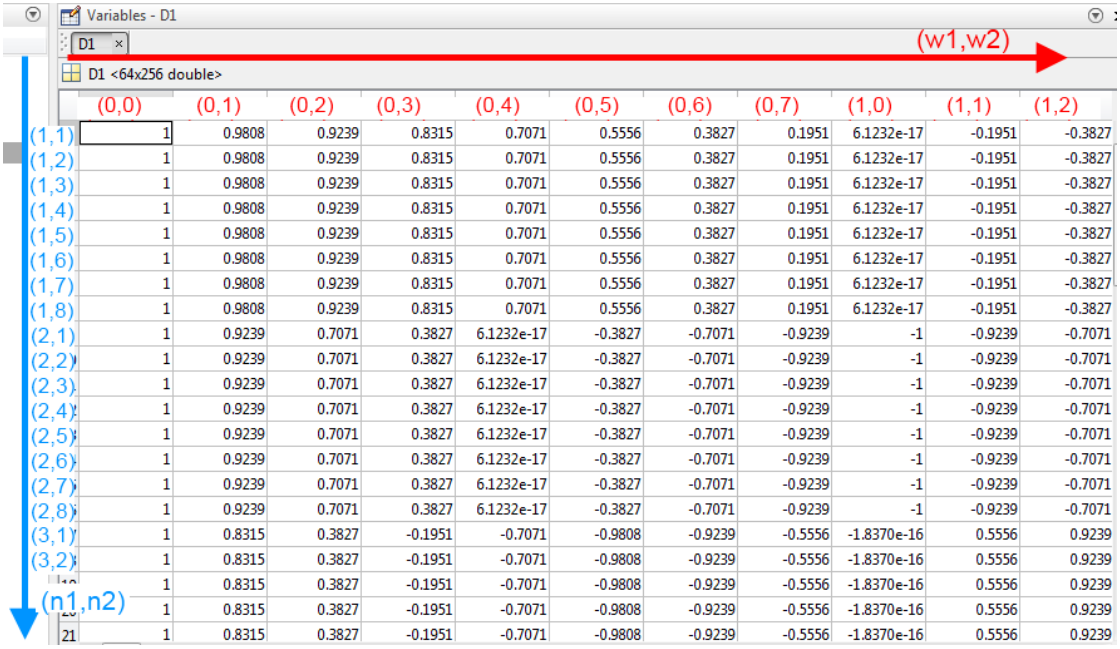
The equation to generate the dictionary is as follows:

$$\cos\left(\frac{\pi}{N}N_1 n_1\right)\cos\left(\frac{\pi}{N}N_2 n_2\right) \quad for\ n_1, n_2 = 1,2,3,\dots,N \tag{6}$$

where $N = 8$ and N_1 and N_2 are integers from 0 to N-1.

As an example which has been implemented in the project, the dictionary uses the following arrangement shown in Figure 2.3 where each column contains 8x8 samples for a specific frequency. The 2D sample ranges from $n_1 = 1,\ n_2 = 1$ to $n_1 = 8, n_2 = 8$. One of the frequency samples, n1 is held constant while the second frequency sample is increased from 1 to 8. Then the frequency sample n1 is increased by 1 until 8 and an increase for n2

from 1 to 8 is repeated, respectively. The 2D frequency ranges of both ω_1 and, ω_2 range from 0 to 3.6Hz. Similar to the frequency samples, one frequency ω_1 is held constant while ω_2 is increased by a 0.4Hz until it reaches 3.6Hz, then ω_1 is increased by 0.4Hz and ω_2 repeats the previous process starting with 0Hz to 3.6Hz.



	(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	(1,0)	(1,1)	(1,2)
(1,1)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(1,2)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(1,3)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(1,4)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(1,5)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(1,6)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(1,7)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(1,8)	1	0.9808	0.9239	0.8315	0.7071	0.5556	0.3827	0.1951	6.1232e-17	-0.1951	-0.3827
(2,1)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(2,2)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(2,3)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(2,4)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(2,5)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(2,6)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(2,7)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(2,8)	1	0.9239	0.7071	0.3827	6.1232e-17	-0.3827	-0.7071	-0.9239	-1	-0.9239	-0.7071
(3,1)	1	0.8315	0.3827	-0.1951	-0.7071	-0.9808	-0.9239	-0.5556	-1.8370e-16	0.5556	0.9239
(3,2)	1	0.8315	0.3827	-0.1951	-0.7071	-0.9808	-0.9239	-0.5556	-1.8370e-16	0.5556	0.9239
	1	0.8315	0.3827	-0.1951	-0.7071	-0.9808	-0.9239	-0.5556	-1.8370e-16	0.5556	0.9239
	1	0.8315	0.3827	-0.1951	-0.7071	-0.9808	-0.9239	-0.5556	-1.8370e-16	0.5556	0.9239
21	1	0.8315	0.3827	-0.1951	-0.7071	-0.9808	-0.9239	-0.5556	-1.8370e-16	0.5556	0.9239

Figure 2.3: PI-DCT sample arrangement

One of the biggest limitations that the OMP algorithms suffers is that the PI-DCT dictionary it uses has certain discrete values from which it can choose basis functions. Thus, regardless of the input image the standard available dictionary is always the same. This makes it simpler to have arrangements in communication systems but it severely limits the ability of the OMP algorithm to represent complex input images.

Thus, for any given image the OMP can only represent the image to a certain degree of accuracy; this accuracy may also vary depending on what kind of frequencies are

present in the image. If the image consists of a large number of high frequencies then their representation becomes harder for a fixed DCT dictionary and the image shows large amounts of blur and overlap in sections of the image where the frequency shifts a lot. The result is usually image dependent and is usually not guaranteed by any algorithm.

This limitation on accuracy can be considered as acceptable if the efficiency of the algorithm is good. Most sparse representation techniques generally find themselves at a loss when it comes to efficiency because the calculation of basis functions is a computationally expensive process. As mentioned earlier, the processing time is limited by getting the algorithm to choose less number of coefficients per window. Accuracy and processing time remain the two features of concern in this study.

Chapter 3
Theory

3.1 Top-Down Search Algorithm (TDS)

At the heart of the TDS algorithm is a curve fitting approach which is explained as follows. Given an input vector x, the function of the TDS is to find the optimal linear combination shown below within the sparsity constraints:

$$X = B \cdot S, \tag{7}$$

where S is the weight matrix.

First a column x is taken from the input matrix X, and the objective now is to find a linear representation of x based on the given dictionary B,

$$x_i = \sum_{j=1}^{m} \left(s_j \times b_{ij}\right), \tag{8}$$

where x_i is the i^{th} row value of column x, b_{ij} is the i^{th} row value of column j in dictionary B, and s_j is the corresponding weight, and m is a constant number.

From Eq. (8), the total error function can be represented as,

$$\theta = \sum_{i=1}^{N} [x_i - \sum_{j=1}^{m} (s_j \times b_{ij})]^2, \tag{9}$$

where N is the number of rows of x.

To obtain a representation that is most accurate and close to the original signal, the error between the two values must be reduced as much as possible. Thus, to minimize the total error, the derivative of θ must be taken and made equal to zero.

$$\frac{d\theta}{ds_j} = \sum_{i=1}^{N} 2 \times \left(-b_{ij}\right) \times [x_i - \sum_{n=1}^{m} (s_n \times b_{in})] = 0. \qquad (10)$$

Simplifying the above equation leads us to:

$$\sum_{i=1}^{N} b_{ij} \times x_i = \sum_{n=1}^{m} s_n \times b_{ij} \times b_{nj}\ , \qquad (11)$$

where *i* and *m* are constants.

Finally, the weights b_{ij} can be calculated using a matrix operation. It should be noted that since b_{ij} and x_i are all known; calculation of the matrices on the right hand side can be done in advance. In this paper, the matrix that requires inverse operation in equation (12) is called matrix D, and the column matrix on the right side of matrix D is called matrix DF.

$$\begin{bmatrix} s_1 \\ \vdots \\ s_m \end{bmatrix} = \underbrace{\begin{bmatrix} \sum_{i=1}^{N} b_{i1} \times b_{i1} & \cdots & \sum_{i=1}^{N} b_{i1} \times b_{iN} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^{N} b_{i1} \times b_{iN} & \cdots & \sum_{i=1}^{N} b_{iN} \times b_{iN} \end{bmatrix}^{-1}}_{\text{Matrix D}} \times \underbrace{\begin{bmatrix} \sum_{i=1}^{N} b_{i1} \times x_i \\ \vdots \\ \sum_{i=1}^{N} b_{im} \times x_i \end{bmatrix}}_{\text{Matrix DF}} \qquad (12)$$

Thus, using the above equations the weights for the given dictionary can be calculated. Initially the TDS algorithm takes all available dictionary atoms, and the entire exercise is to

reduce the number of atoms to represent the input signal within the sparsity constraints. This is done in an iterative manner by using the curve fitting approach to calculate the weights for the current active dictionary atoms and deactivate one single atom from the active dictionary atoms which gives the least error increase. In each iteration, an active atom is removed once to compute all possible representations exhaustively. For instance, if there are n active dictionary atoms, then there are totally n possible representations. Figure 3.1 gives an outline of the steps involved in the TDS algorithm.

Top − Down search algorithm

1 $S = TDS(B, x, sparsity)$
2 *calculating the matrixes in equation* (1) *in advance with* m = *number of rows in* B
3 *while not finished*
4 $m = m - 1$
5 *while* $i <$ *number of columns in* B
6 D_i = *remove* b_{in} *from* B
7 *calculating the weight vector* s *based on* D_i
8 $error_i = ||x - D_i \cdot s||$
9 *end*
10 *find* $i: |$ *minmum error* $= min(error_i)$
11 $B = D_i$
12 *if sparsity = constraint then finish*
13 *end*
14 *return*

Figure 3.1: TDS algorithm

The error difference between these representations and the original signal are calculated and saved. With the list of errors; the algorithm finds the representation which gives the least possible error and permanently deactivates the corresponding atom. This new active dictionary now replaces the previous active dictionary and process repeats itself

collapsing the dictionary as it goes. The algorithm finishes when the sparsity constraint is achieved. This collapse effect is carried out until the required sparsity factor is attained. This guarantees that the result has the minimal possible error. However the number of computations to get to this accurate result is extremely high making the processing efficiency of the algorithm low. The results of the comparison between TDS and OMP are presented in Chapter 6.

3.2 Properties of The Dictionary

The properties of the dictionary prove to be crucial when representing images. As explained earlier depending on what atoms are present in the dictionary the resulting representation maybe either highly accurate or moderately accurate. For example if the input image to the system has all low frequency components and the dictionary being used contains all low frequency components then there is a high probability that sparse coding algorithms used will be able to find an accurate representation for the input image. Even with all the low frequencies present in the dictionary there is a chance that the representation is not as accurate as required because of the discrete nature of the DCT dictionary, therefore it is difficult to account for frequencies that fall in between the chosen frequencies inside the DCT dictionary. Thus, there is always a trade off when it comes to sparse representations and much research has been done into how to make a balanced general dictionary that can account for images with different frequency components. A

number of dictionary properties are investigated in this thesis. The key features are described below.

(1) Frequency Modification

The standard PI-DCT dictionary follows the equation that $w = \frac{N1}{N}\pi$ $N1 = 0,1,2 \ldots N-1$, where N is number of pixels in the image. *N1* uses evenly distributed discrete values. As mentioned earlier, this becomes a problem because the global optimum frequencies needed to accurately represent the input image may land somewhere in between two discrete frequency values of the DCT dictionary. For example from the graph in Figure 3.2 it is observed that the intensity of the signal varies

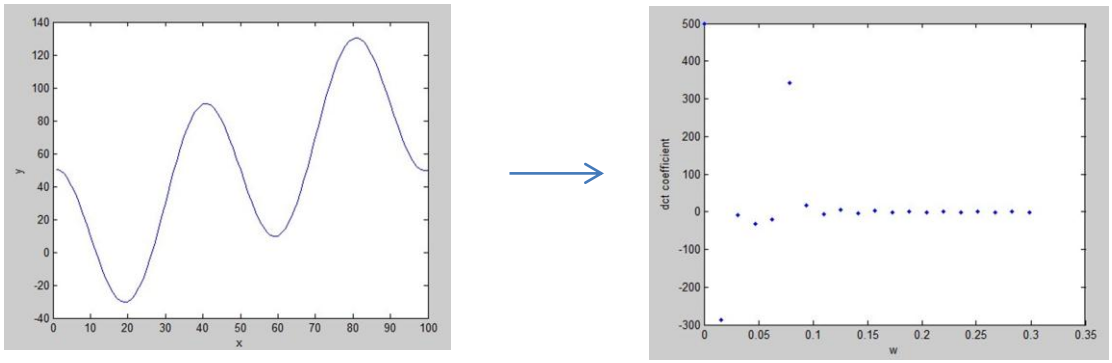


Figure 3.2: Original signal and frequency domain after discrete cosine transform

frequency domain with discrete values the frequency domain representation may not be adequately represented. This happens because the frequencies chosen to be added to the dictionary do not contain enough detail to correctly represent the details of the original image.

The question arises that if the standard PI-DCT dictionary fails to adequately represent the signal required in the frequency domain; what can be done to improve the accuracy of the representation. The first obvious answer that comes to mind is; why not increase the number of dictionary elements. This concept works and is referred to as an "over complete dictionary" where the number of atoms in the dictionary are much greater than the number of atoms defined by the sparsity factor. Although an increased dictionary size does not increase the computational time a lot for the OMP algorithm when finding the solution for one single input. It is still necessary however to keep the dictionary size as small as possible to increase the efficiency of the system when the OMP is applied on a real image.

The natural question that arises next is that whether there is any way to intelligently expand the dictionary around the frequencies that are necessary. This would allow for only a moderate expansion in the size of the dictionary allowing better results with increased efficiency. The technique proposed in this thesis is to first use the OMP algorithm to find a representation for the input signal with a specific sparsity factor. Once this is done and the required dictionary atoms are found, the result can be fine-tuned by searching around these atoms for better alternative frequency values. For example if the OMP algorithm chose 2π and 3π originally, then the dictionary can be expanded in between these two individual frequencies to include 2.1π, 2.2π, 2.3π, 2.4π, 2.5π, etc., and the OMP algorithm is run again. In theory the expanded number of options will remove

any constraints of the DCT dictionary which are present in the original dictionary thus giving more accurate results. This method is referred to as intelligent expansion. The results of evenly expanded dictionary and the intelligently expanded dictionary are described in Chapter 6.

(2) Phase Shift

The original PI-DCT dictionary does not offer all elements to represent the entire signal space for image construction. A PI-DCT dictionary with phase shifting elements has the possibility to represent more elements in the signal space. Therefore, the PI-DCT dictionary can be expanded with phase shifting to increase accuracy for image signal representation.

The following equation is the original PI-DCT representation function with a phase-shift of zero for the domain $0 \leq \varphi < 2\pi$ where φ is the phase shift:

$$\cos\left(\frac{\pi}{N}N_1 n_1\right)\cos\left(\frac{\pi}{N}N_2 n_2\right) \qquad for\ n_1, n_2 = 1,2,3,\dots,N \tag{13}$$

where N_1 and N_2 are from 0 to N-1.

The equation can be modified to allow phase shift and is defined as:

$$\cos\left(\frac{\pi}{N}N_1 n_1 + \varphi_1\right)\cos\left(\frac{\pi}{N}N_2 n_2 + \varphi_2\right) \quad for\ n_1, n_2 = 1,2,3,\dots,N \tag{14}$$

where N_1 and N_2 are from 0 to N-1 and φ_1 and φ_2 are the phase shifts for the cosine functions 1 and 2 used to construct the PI-DCT dictionary. Both phases φ should be bounded by $0 \leq \varphi < 2\pi$ to prevent repetition of elements in the PI-DCT dictionary.

Similar to frequency expansion, phase shift expansion can be expanded evenly or intelligently. Each element in an un-expanded phase shift dictionary can be expanded with an unlimited number of phases. That number depended on the precision desired for the problem. Similar to the frequency expansion, the number of phases cannot be predicted and a threshold may exist.

(3) PSO Algorithm

In the previous two approaches, the dictionary is expanded based on the standard DCT dictionary; however the solution space of finding the best dictionary is obviously non-linear as shown in figure 3.3, therefore simply expand the DCT dictionary is not

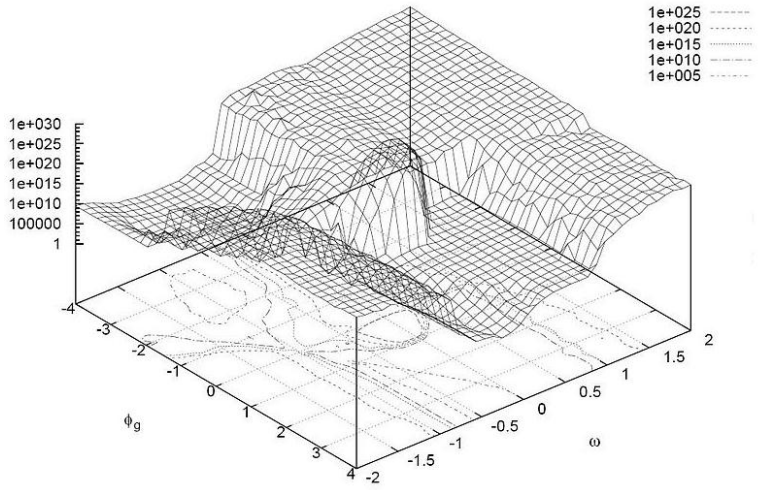


Figure 3.3: Non-linear solution space

capable of finding the global optimum solution. Instead of being restricted by the frequency and phase of DCT dictionary, a random search of the solution space using Particle Swarm Optimization (PSO) algorithm has a number of advantages: (1) it always

aims for the global optimum, (2) it usually converges to a good solution very fast, and (3) it is easy to modify to meet different constraints such as sparsity. The downside of heuristic approach when applied to dictionary learning is that it may take too long time to reach a solution which is much better than the solutions achieved by the algorithms described in the previous sections because the solution space is too big. How the PSO algorithm being used performs depends largely on whether the heuristic algorithms' coefficients are well tuned, and the trade-off between convergence speed and quality of solutions.

Given the time limitations of this project, only one heuristic algorithm, Particle Swarm Optimization algorithm (PSO) is implemented. PSO is a type of random parallel optimization algorithms. It is originally designed to for simulating social behavior, as a stylized representation of the movement of birds. PSO optimizes a problem by having a population of candidate solutions called particles to move around the solution space, and trying to move to a better position by considering the current best position of each single particle and the current best position among all particles. It has the advantage of: (1) no strict requirement of the objective function such as the objective function has to be differentiable and continuous, (2) it usually converge faster than other heuristic approaches, (3) very easy to implement using any programming language. However, PSO also has the disadvantage of: (1) not working very well with the objective function which has a number of local optimum solutions, (2) it usually cannot achieve a highly accurate

solution without the help of precision search algorithm, (3) there is no guarantee that the algorithm will converge to a global optimum.

Chapter 4
Simulation

4.1 Assumption

In the course of the testing process a number of assumptions are made to simplify the process and standardize the results across the different domains of testing. The assumptions are listed as follows:

1 A portion of the standard Lena image is used for the input for the tests as shown in figure 4.1. For the majority of testing an eye portion of the Lena image was used.

Figure 4.1: Standard image test picture Lena

2 The OMP algorithm is used for testing the dictionary parameters through all the different variations; i.e., Frequency variations and Phase variations.

3 In all the tests (except the modified dictionary tests) a standard PI-DCT dictionary is used.

4 To judge the results of each algorithm and variation the absolute error equation is used which is given by

$$Error = \sqrt{\sum_{n=1}^{N}(x_n - x_n')^2} \quad (15)$$

The unit of the error is gray scale level in this project since the picture is black and white.

4.2 Top-Down Algorithm Simulation Flow

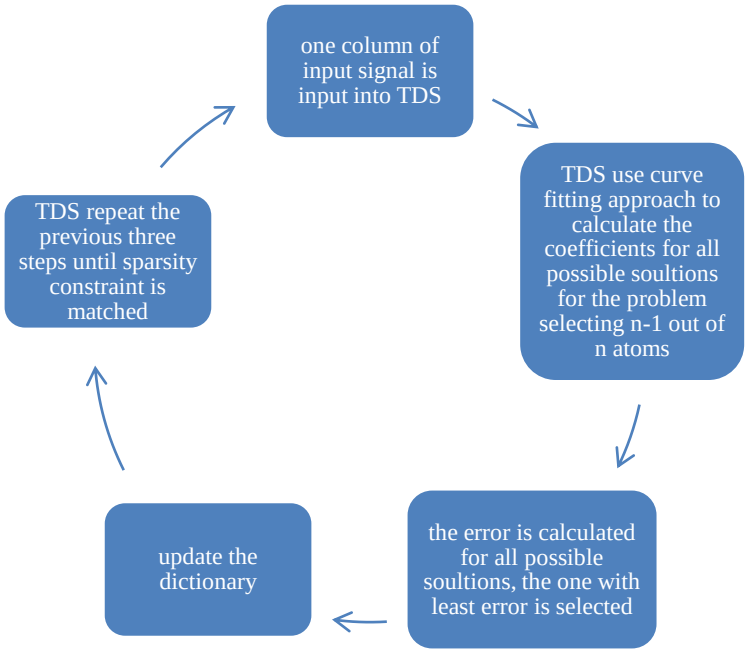


Figure 4.2: Top-Down search algorithm flow chart

Figure 4.2 represents the flow of the TDS algorithm simulation. The whole program is implemented using MATLAB language. As illustrated in the assumption section, a

portion of a standard Lena picture is selected and divided into 8x8 blocks. Before the data is input into TDS, all these 8x8 blocks are reshaped to 64x1 columns. TDS processes each column of input signal separately and starts by using the curve fitting approach illustrated in Chapter 3.1 to calculate the coefficients for all possible solutions. Notice that the matrix D and matrix DF required by curve fitting in Chapter 3, Eq. (12) are calculated in advance out of TDS function. Therefore for each solution of selecting n-1 atoms out of n atoms, the TDS function simply removes the corresponding row and column from matrix D, and then removes the corresponding row from matrix DF, and then finally calculating the coefficients using Eq. (12) in Chapter 3. TDS repeat this optimization process until the number of selected atoms matches the sparsity constraint, and then it start processing the next input column. Finally, TDS finish execution after processing all input columns.

4.3 Modification of The Dictionary

(1) Even Expansion of The Standard PI-DCT Dictionary

The equation for the NxN PI-DCT dictionaries is

$$cos\left(\frac{\pi}{N}(N_1)n_1\right) cos\left(\frac{\pi}{N}(N_2)n_2\right) \; for \; n_1, n_2 = 1, 2, 3, \dots, 8 \qquad (16)$$

To generate this dictionary in MATLAB, two "for" loops are used to produce the DCT table. The number of iterations in each loop depends on the frequency range for $\omega_1 = \frac{\pi}{N}(N_1)$ and $\omega_2 = \frac{\pi}{N}(N_2)$.

Increasing the number of iterations in both for loops by a factor of 2 creates an evenly expanded dictionary. The terms inside both for loops are also divided by the same factor to maintain the same frequencies for both ω_1 and ω_2.

The following equation is coded in MATLAB where N is a positive integer that is a factor of 2.

$$cos\left(\frac{\pi}{2 \times N}(N_1)n_1\right) cos\left(\frac{\pi}{2 \times N}(N_2)n_2\right) \quad for\ n_1, n_2 = 1,2,3, \dots ,8 \qquad (17)$$

For example, to create an evenly spaced PI-DCT dictionary with one additional frequency between two frequencies would produce a 64x256 PI-DCT dictionary. Each loop is modified to N1=0 to 15 and N2=0 to 15 and that the number inside the cosine terms are divided by 2. In MATLAB, all N^2 values of one atom which has fixed ω_1 and ω_2 is saved using column matrix with size of N^2x1.

(2) Intelligent Expansion

The second strategy of expanding the dictionary is based on the idea that instead of expanding the PI-DCT dictionary uniformly every time with even space among all frequencies of the dictionary atoms; why not to expand the dictionary around the best

atoms for a better result, in other words a precision enhancing technique. To do this the OMP algorithm is first run using the standard PI-DCT dictionary and allowed to choose the best result based on the sparse constraints. Based on this the frequency domain around the frequencies (which are selected by OMP algorithm originally) are searched iteratively with increasing precision, allowing arbitrary frequencies in between the frequencies of standard PI-DCT dictionary to be reached.

$$cos\left(\left(\frac{\pi}{N}N_1 + precision * m\right)n_1\right) cos\left(\left(\frac{\pi}{N}N_2 + precision * m\right)n_2\right) \quad (18)$$

$$for\ n_1, n_2 = 1,2,3,\dots,8$$

Where N_1 and N_2 are integers ranged from 0 to N-1.

The precision of the dictionary depends on the separation of the atoms between the standard frequency values. Thus, if a greater deal of accuracy is needed then successive frequencies should be more finely separated. For example successive separations of 0.11, 0.12, and 0.13 have greater precision than 0.1, 0.2 and 0.3.

The technique used in the testing of the intelligent dictionary uses an increasing precision concept where the value of the precision is divided up successfully. So for example a frequency of 2Hz is being searched around the answer. The first sets of atoms searched are 2.1Hz, 2.2Hz and 2.3Hz. Once the best atom is chosen from these options the search is continued around the same atom. So for example if 2.2Hz is chosen, the algorithm searches around 2.21 Hz, 2.22 Hz and 2.23 Hz etc. Thus, the precision value decreases by a dividend of 10 each time.

Lastly, when exploring values around a particular frequency to increase its precision it is necessary to explore possible atoms on both sides of the value. For example if there is a frequency of 1Hz All atoms around it must be considered. The range of possible solutions therefore considered would 0.1 to 0.9 and 1.1 to 1.9.

After frequency modification, the dictionary then goes through the phase shift. The phase shift simply follows the following equation.

$$cos\left(\left(\frac{\pi}{N}N_1\right)n_1 + phase \times a\right) cos\left(\left(\frac{\pi}{N}N_2\right)n_2 + phase \times b\right), \quad (19)$$

$$a\ and\ b = 1,2,3,4\ \ldots$$

In the above equation, a and b are defined by user depends on how much phase shifts are required. The result of smart expansion is discussed in detail in Chapter 5.2.

(3) PSO Approach

Instead of expanding the standard DCT dictionary on the frequency domain, the PSO algorithm simply does a random search of the whole solution space with certain rules. In this project, PSO is implemented to check if the PSO approach has the ability to provide good solutions in a more efficient manner. The simulation flow is illustrated in the following paragraph.

The OMP algorithm is used to calculate the coefficients for all solutions, and then the objective function, which is the total error function shown in Chapter 4.1, is

calculated using the coefficients from OMP algorithm. The objective is to find the dictionary which provides the least total error.

In the MATLAB implementation, the initialization is done at the beginning as follows; $m = 50$ is the total number of particles in the swarm, each having a position $x_i \in \mathbb{R}^n$ in the search-space and a velocity $v_i \in \mathbb{R}^n$ indicating the speed, and p_i is the best known position of particle i and g is the best known position of the entire swarm.

For each particle $i = 1, \ldots, m$, it first initialize the particle's position with a uniformly distributed random vector: $x_i \sim \mathrm{U}(b_{lo}, b_{up})$, where b_{lo} and b_{up} are the lower and upper boundaries of the search-space. After the initial position is generated, the particle's best known position is calculated by the objective function, and it is saved into p_i: $p_i \leftarrow x_i$, and then the best known position of particle p_i is compared to the current best known position of the entire swarm g, if any p_i provides a better result, then the swarm's best known position is updated: $g \leftarrow p_i$. Following the previous steps, the particle's velocity is initialized: $v_i \sim \mathrm{U}(-|b_{up} - b_{lo}|, |b_{up} - b_{lo}|)$. Finally the program simply repeats the following steps until a termination criterion is met:

- Pick random numbers: $r_p, r_g \sim U(0,1)$, and then update the particle's velocity using the following equation: $v_{new} = \omega \times v_{i,d} + c_1 \times r_p \times (p_{i,d} - x_{i,d}) + c_2 \times r_g \times (g_d - x_{i,d})$ for each particle, where ω is the inertia coefficient, and d is the dimension.

- Update the particle's position: $x_i \leftarrow x_i + v_i$.
- Update the particle's best known position if better position is found: $p_i \leftarrow x_i$
- Update the swarm's best known position if the current best position among all particles is better than g: $g \leftarrow p_i$.

  The best position of swarm g always holds the best solution among all trails.

Chapter 5

Results and Discussions

5.1 TDS vs. OMP

When the 64x64 non-expanded PI-DCT dictionary with evenly spaced frequencies is used with both the OMP and TDS algorithms, it is observed that neither of the approaches is able to acquire a global optimum value. The test image used in this test is the eye portion of the standard black and white Lena image for DSP application testing.

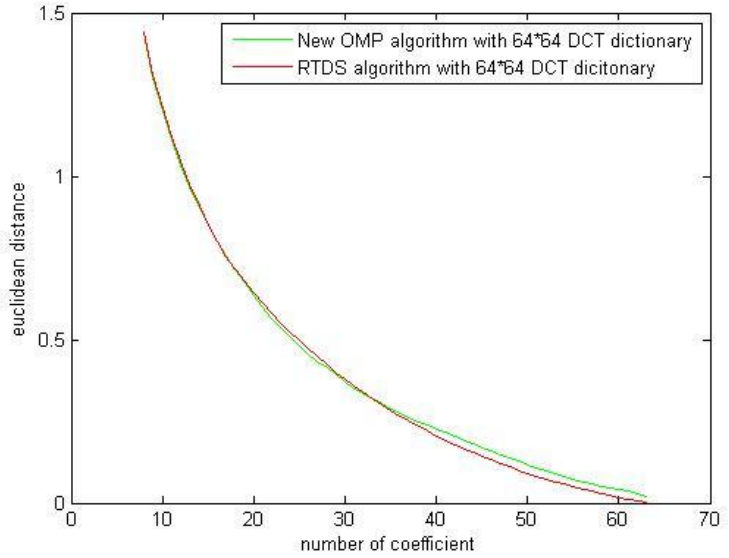


Figure 5.1: OMP vs. RTDS using the standard 64x64 PI-DCT Dictionary

In the Figure 5.1, OMP shows very similar results compare to RTDS when the image is reconstructed with low sparsity of between 10 to 20 out of 64 coefficients. However,

with 20 to 35 coefficients, the results for OMP are slightly better than RTDS. For over 35 coefficients, TDS shows better performance over OMP.

(a) (b) (c)

Figure 5.2: (a) Original picture, (b) Reconstructed picture using OMP with sparsity of 8, (c) Reconstructed picture using Top-Down search algorithm

Figure 5.2 shows the results of the different runs that were conducted. Figure 5.2 represents the original image which is given as input to the system. Figure 5.2b represents the recreated representation of the input image using OMP with a sparsity of 8. Finally Figure 5.2c is the recomposed image using the TDS approach. The results have proven that both methods do not form the global optimum because the results for both methods intersect and cross each other at different sparsity. A global optimum method will result in lower error through the entire sparsity range.

Both methods cannot reach the global optimum due to the fact that both algorithms make the assumption that the first element selected is the best element to be selected in the entire dictionary leading to the optimum result. There is a difference in approach between OMP, which selects elements from the dictionary to form a representation set and the RTDS which is initialized with all elements in the representation set and elements are one

by one discarded in each iteration. Each element removed has a direct link to the previous iteration leading to a dependency on the first element selected or removed. Both algorithms would only know the optimum for the next best result but not the optimum results beyond the next iteration.

From Figure 5.1, it is observed that when the sparsity is high, TDS performs better than OMP. To test how much error reduction the TDS achieve when the sparsity is really high, a dictionary with size of 256x256 is used in the next test. The dictionary is generated using the same equation in Chapter 4, the only difference is n1 and n2 range from 0 to 1 to 16. The result is shown in Figure 5.3 and Figure 5.4. It turns out that when the sparsity is really high, OMP's performance is worse than TDS.

sparsity	200	201	202	203	204	205	206	207	208	209
OMP(PSNR)	47.73dB	47.87dB	48.04dB	48.20dB	48.36dB	48.53dB	48.76dB	48.95dB	49.12dB	49.31dB
TDS(PSNR)	52.73dB	52.80dB	52.85dB	53.02dB	53.18dB	53.32dB	53.45dB	53.51dB	53.67dB	53.83dB

Table 5.1: Error of OMP vs TDS using 256x256 dictionary

(a) (b) (c)

Figure 5.3:(a) Original picture, (b) Reconstructed picture using Top-Down search algorithm with sparsity of 200, (c) Reconstructed picture using OMP with sparsity of 200

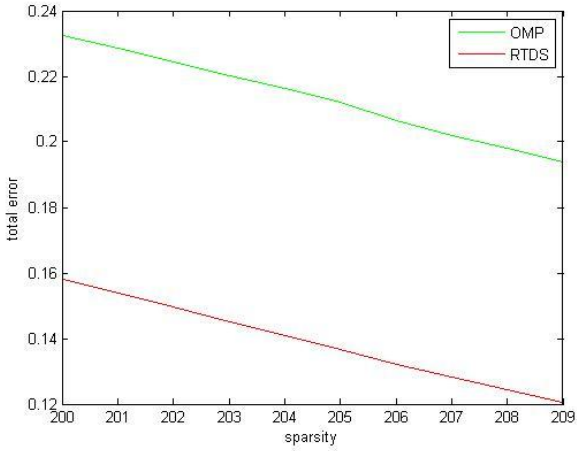


Figure 5.4: OMP vs TDS using 256x256 dictionary

The efficiency of OMP algorithm is that it does not perform excessively exhaustive searches resulting in much quicker execution time for a good solution. The disadvantages are that it does not achieve the global optimum, and may sometimes get a worse solution when the dictionary size is increased. Secondly when the sparsity is high OMP's performance is worse than TDS. The second downside of OMP is explained in detail in Chapter 5.2. TDS uses curve fitting to calculate the coefficient for selected dictionary, so it has higher accuracy than OMP in terms of coefficient calculation, and it works very well when the sparsity is high. However, it still cannot reach the global optimum. Another downside is that TDS cannot be directly applied on over-complete dictionary, whose column number is bigger than row number, because this prohibits the curve fitting approach from finding a unique solution.

Although both algorithms result in very similar result, OMP is much better in execution time for low sparsity representation because there are much less iterations to

calculate from bottom up than from top down, while TDS is slightly better than OMP in terms of overall error.

5.2 Dictionary Expansion Strategies Comparison

One of the biggest advantages of dictionary expansion is that it can be used to evenly expand the dictionary to get more precise elements; this allows the OMP to minimize the error significantly while maintaining the same sparsity. This is a major advantage because if it is applied to data transmission this would mean that the transmission data size remains the same as using a smaller dictionary.

The disadvantage of dictionary expansion is that producing the PI-DCT dictionary is a very heavy computational process because each element is computed by multiplication between two cosine functions. It also takes 64 times longer to produce the 64x4096 dictionary. Although this process is very computationally expensive and consumes many resources, this process only occurs when the dictionary is not available. Furthermore, the amount of random access memory (RAM) required for computing the dictionary increases exponentially with respect to the dictionary size. The receiver has the advantage of receiving the dictionary from the sender instead of computing it again.

Another disadvantage of expanding the dictionary is that the size of the dictionary expands exponentially. The 64x4096 evenly expanded dictionary is 2^6 or 64 times larger. The sender and receiver need to keep a copy of this large dictionary to encode and decode

the sparsity data being transmitted. The final disadvantage of the expansion is that the OMP execution time is slightly longer compared to the original 64x64 dictionary because there are more elements for OMP to compare with but the sparsity did not change.

As discussed earlier there are three main approaches inside the dictionary expansion that are discussed in this thesis, namely, the even expansion, the intelligent expansion and PSO approach. From the study it is concluded that regardless of what method is used dictionary expansion always allows the OMP to find better solutions than non-overcomplete dictionary, because it has a larger solution space and can find better answers. So compared to the standard 64x64 DCT dictionary the expanded dictionaries always provides a better solution. The efficiency in both cases however goes down due to increasing size of the dictionary.

Size of Dictionary	64x64	64x256	64x1024	64x4096	64x16384
Even expansion	30.82dB	31.01dB	31.15dB	30.87dB	30.68dB
Smart expansion	31.03dB (64x445)	30.42dB (64x501)	30.78dB (64x1005)	31.14dB (64x1061)	30.48dB (64x1229)
PSO algorithm	29.57dB	30.05dB	30.68dB		

Table 5.2: Comparison between evenly expansion, smart expansion, and PSO (SNR)

An important result that should be pointed here is that as the size of the dictionary increases with the intelligent dictionary the results has no improvement, and even degrade as shown in Table 2. This does not come as a glitch and successive trails led to the same answer. The cause of this occurs from the fact that the OMP algorithm is originally a greedy algorithm; thus when provided with the choice of an atom at a particular stage it chooses the best one regardless of the consequences of the future. Thus choosing a better atom at an earlier stage does not guarantee that the eventual outcome will be a highly accurate result, even though the OMP considers its future answers based on past choices. There is reason to suggest selection of a good atom puts the system on a particular branch of solutions. This could lead to a very accurate result or a result that is slightly worse compared to other dictionary sizes. In this case the solution chosen was worse than the previous options available.

PSO algorithm is proved to be able to provide good solutions. Compared with even expansion and intelligent expansion, the error is really close but still a bit worse. In this project, only 10 iterations are tested, the reason is in the objective function, OMP algorithms has to be used to calculate the coefficient in each iteration, therefore the PSO approach spend too long time on the objective function. The convergence speed is shown in Figure 5.5. To reach a better result, the PSO algorithm should take more iteration, and all the PSO coefficients have to be tuned properly. The reason that there is no data for 64x4096 and 64x16384 is it takes too long time to finish the execution because the dimension of PSO is 4096 and 16384.

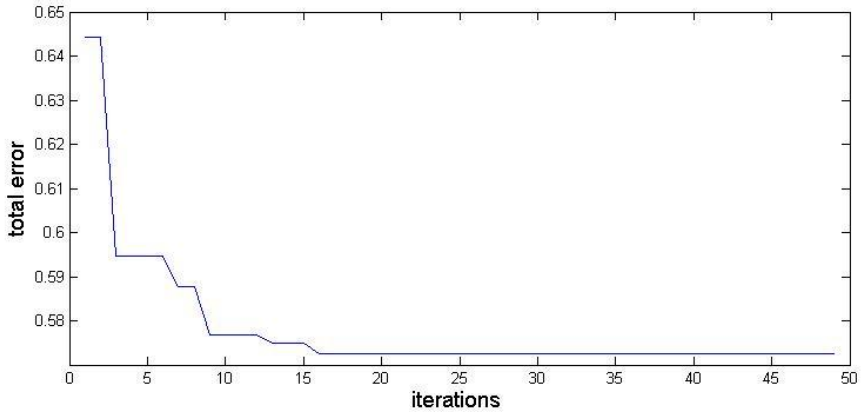


Figure 5.5: PSO convergence speed

The study concludes that the evenly expanded dictionary gives a slightly better result compared to the intelligently expanded dictionary and PSO approach as is visible from Table 2. This happens for a number of reasons. Firstly, when frequencies are chosen using the evenly expanded dictionary the solution space to choose from is wider. Thus, even if the OMP cannot find us a global optimum; it can use curve fitting techniques in a better way and reach a near optimal solution. In the Intelligent search the OMP first picks the best available frequencies to represent the input image. Next the frequency range around these particular frequencies is searched to see if a better result can be located. This is done under the assumption that it is possible due to the discrete values of the dictionary the OMP can only pick what is available to it in the initial run, thus, missing the opportunity to find a better solution that could be available if other frequencies could be explored in the area between the discrete values in the dictionary. Thus, if the solution curve follows a trend searching around the area of these originally picked frequency values should give a better result. However this method has not generated better results than the evenly expanded

dictionary; while the error in both techniques is lower, the evenly expanded dictionary gives a slightly better result. The problem with the intelligent search method is that because it focuses on expanding around the initially picked solutions it doesn't necessarily mean a global optimum value can be found. This is because the solution space varies and doesn't follow a constant trend. Expanding around the original frequencies may lead to worse solutions or only slight variations in error. In most cases it was observed that when an enlarged dictionary from the intelligent search was used; the OMP chose to keep its original choices and not change too much after the expansion. Expanding around the original choices also creates a problem in that the flexibility of the system is reduced. Thus it is concluded that while dictionary expansion of any nature will give a naturally better result when compared to a standard DCT dictionary, it is better to use an evenly expanded dictionary and not the intelligently expanded dictionary whose original motivation was to increase the precision of the result but does not bring any significant drops in error.

Chapter 6
Conclusion and Future Work

6.1 Conclusion

In this project a number of assessments were made about the popular OMP algorithm. OMP is able to provide good results under a given set of sparsity constraints while being able to maintain its efficiency, but the error can be reduced further through other techniques such as TDS. OMP has a downside that has not been observed before; the error may increase even if a bigger size dictionary is used. The reason is that OMP is essentially a bottom-up algorithm, and it always keeps the residue orthogonal to the current selected dictionary atom as explained in Chapter 5.2.

The even dictionary expansion proves to be the most efficient way among all three approaches, while the smart expansion has several downsides: (1) not as efficient as even expansion, (2) may cause OMP algorithm to get even higher error after dictionary is expanded, (3) the expansion result varies a lot depending on the sparsity of the initial solution. For the heuristic algorithm approach, the performance of PSO is not as good as the even expansion and intelligent expansion approaches mainly because the solution space is too big, and there is no high precision algorithm to assist. A better result may be achieved by tuning the coefficients of PSO properly and increasing the number of iterations. It is also observed that the atoms selected by the sparse coding algorithm share a

dependency which is why the performance of the OMP may degrade at times even though the dictionary size is increased. Thus, heuristic algorithms like Genetic Algorithm may perform better than PSO and need to be tested extensively to see if a heuristic approach can perform better than the dictionary expansion approaches.

The ability to have known dictionaries is a very important step in the application of this study to communication systems. When large amounts of data need to be transmitted across lines between a transmitter and a receiver; sparse representations prove valuable as their compressed form allows for smaller amount of data to be sent between transmitter and receiver and a reconstruction of the input to be reproduced at the receiver.

6.2 Future Work

There are numbers of possible extension to this project:

1. TDS proved that the error can be further reduced, however how to make this approach more practical in terms of execution time needs to be studied in more detail.
2. TDS can be applied using over-complete dictionary with the assistance from OMP.
3. Instead of exhaustively search a small solution space as what we did in TDS, heuristic algorithm may be applied to achieve a more efficient partial search.
4. Hybrid heuristic algorithms such GA/PSO may achieve better result than PSO. A greedy algorithm can be used to provide better initial solutions.

Bibliography

[1] D. L. Donoho, "Compressed Sensing," *IEEE Transactions on Information Theory,* vol.52, no. 4, pp.1289-1306, 2006.

[2] M. Elad, and M. Aharon, “Image Denoising Via Sparse and Redundant Representation Over Learned Dictionaries,” *IEEE Transactions on Image Processing*, vol.15, issue.12, pp.3736-3745, 2006.

[3] X.C. Lian, Z.W. Li, C.H. Wang, B.L. Lu, and L. Z, “Probabilistic models for supervised dictionary learning,” 2010 IEEE conference on Computer Vision and Pattern Recognition (CVPR), pp.2305-2312, 2010.

[4] O. Bryt, and M. Elad, "Compression of Facial Images Using the K-SVD Algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, 2008.

[5] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: Design of Dictionaries for Sparse Representation,” *IEEE Transactions on Signal Processing*, vol. 54 , no. 11, pp. 4311-4322, 2006.

[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online Dictionary Learning for Sparse Coding,” *Proceedings of the 26th Annual International Conference on Machine Learning*, pp.689-696, 2009.

[7] K. Skretting, and K. Engan. “Recursive Least Squares Dictionary Learning Algorithm,” *IEEE Transactions on Signal Processing*, vol.58, no.4, pp. 2121-2130, 2010.

[8] R, Rubinstein, A.M. Bruckstein, and M. Elad, “Dictionaries for Sparse Representation Modeling,” *Proceedings of the IEEE*, vol.98, no. 6, pp. 1045-1057, 2010.

[9] M. D. Plumbley, “Dictionary Learning for L1-Exact Sparse Coding,” *ICA'07 Proceedings of the 7th international conference on Independent component analysis and signal separation*, pp. 406-413, 2007.

[10] Q. Liu, S.S Wang, and J.H. Luo, “A novel predual dictionary learning algorithm,” *Journal of Visual Communication and Image Representation*, vol.23, no. 1, pp.182-193, January. 2012.

[11] H. Lee, A. Battle, R. Raina, and AY. Ng, "Efficient Sparse Coding Algorithms," *Advances in Neural Information Processing Systems (NIPS)* 19, 2007.

[12] M. Zheng, J. Bu, C. Chen, and C. Wang, “Graph Regularized Sparse Coding for Image Representation,” *IEEE Transactions on Image Processing*, vol.20, no.5, pp.1327-1336, 2011.

[13] M. Yang, L. Zhang, J. Zhang, and D. Zhang, “Robust Sparse Coding for Face Recognition,” *2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp.625-632, 2011.

[14] K. Skretting, "Partial Search Vector Selection for Sparse Signal Representation," *In Proceedings of IEEE Norwegian Symposium on Signal Processing (NORSIG '03)*, October. 2003.

[15] L. Rebollo-Neira, and D. Lowe. "Optimized Orthogonal Matching Pursuit Approach," *IEEE Signal Processing Letters*, vol.9, no.4 , pp. 137-140, 2002.

[16] G. Rath, and C. Guillemot, “Sparse approximation with an orthogonal complementary matching pursuit algorithm,” *IEEE International Conference on Acoustics, Speech and*

Signal Processing 2009, pp. 3325-3328, 2009.

[17] M. Gharavi-Alkhansari, and T.S. Huang, "A fast orthogonal matching pursuit algorithm," *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing 1998*, vol. 3, pp.1389-1392, 1998.

[18] Z. Hussain, J. Shawe-Taylor, D.R. Hardoon, and C. Dhanjal, "Design and Generalization Analysis of Orthogonal Matching Pursuit Algorithms," *IEEE Transactions on Information Theory*, vol.57, no.8, pp.5326-5341, 2011.

[19] L. Rebollo-Neira, and D. Lowe, "Optimized orthogonal matching pursuit approach," *IEEE Signal Processing Letters*, vol.9, no..4, pp. 136-140, 2002.

[20] N . Ahmed, T. Natarajan, and K.R. Rao, "Discrete Cosine Transform," *IEEE Transactions on Computers*, vol. C-23, no.1, pp.90-93, 1974